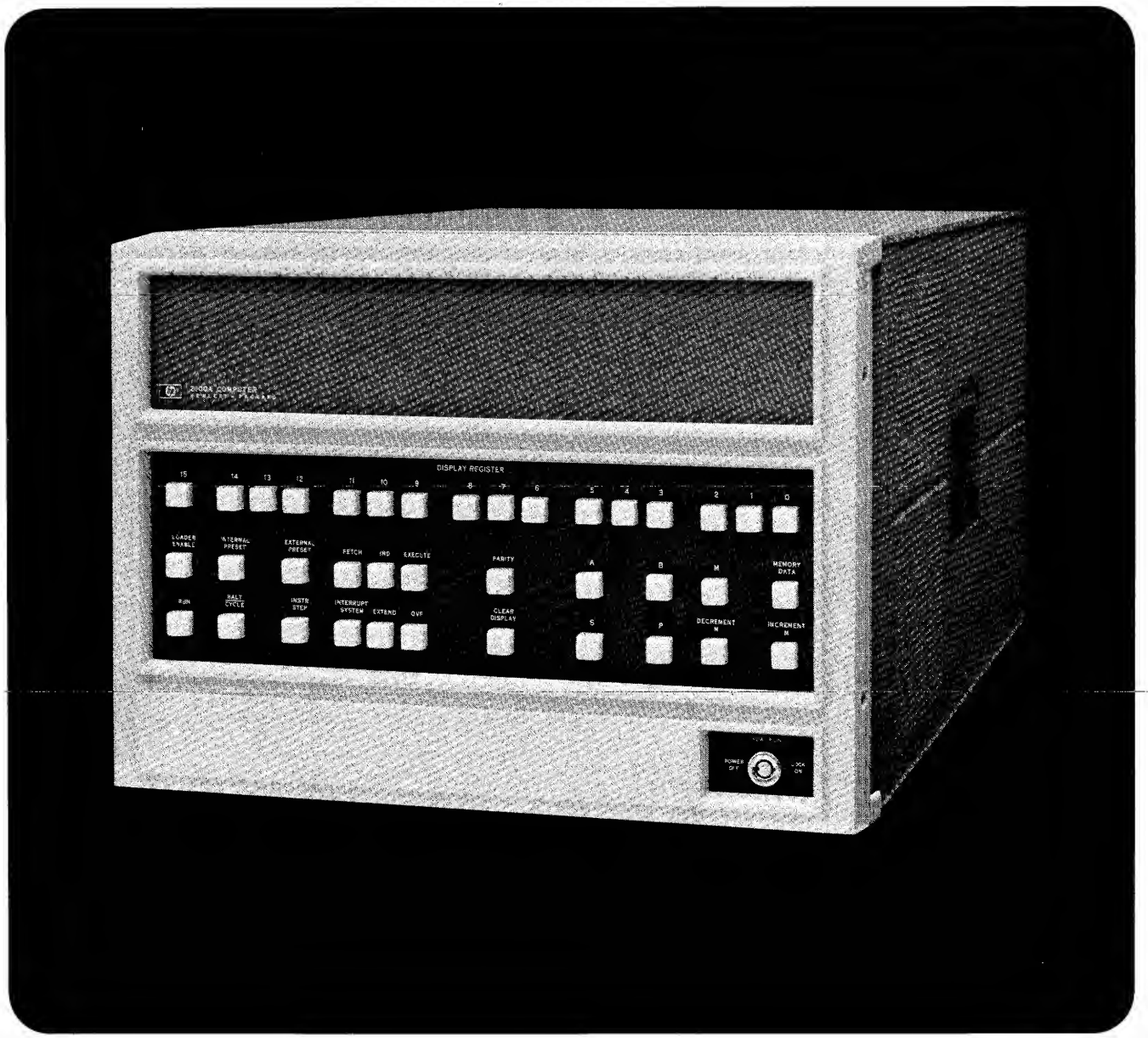
HEWLETT PACKARD

2100A computer



reference manual

HEWLETT hp PACKARD

2100A COMPUTER

REFERENCE MANUAL

02100-90001

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, U.S.A.

Printed: DEC 1971

CONTENTS

ILLUSTRATIONS

TABLES

APPENDIX

ALPHABETICAL INDEX OF INSTRUCTIONS

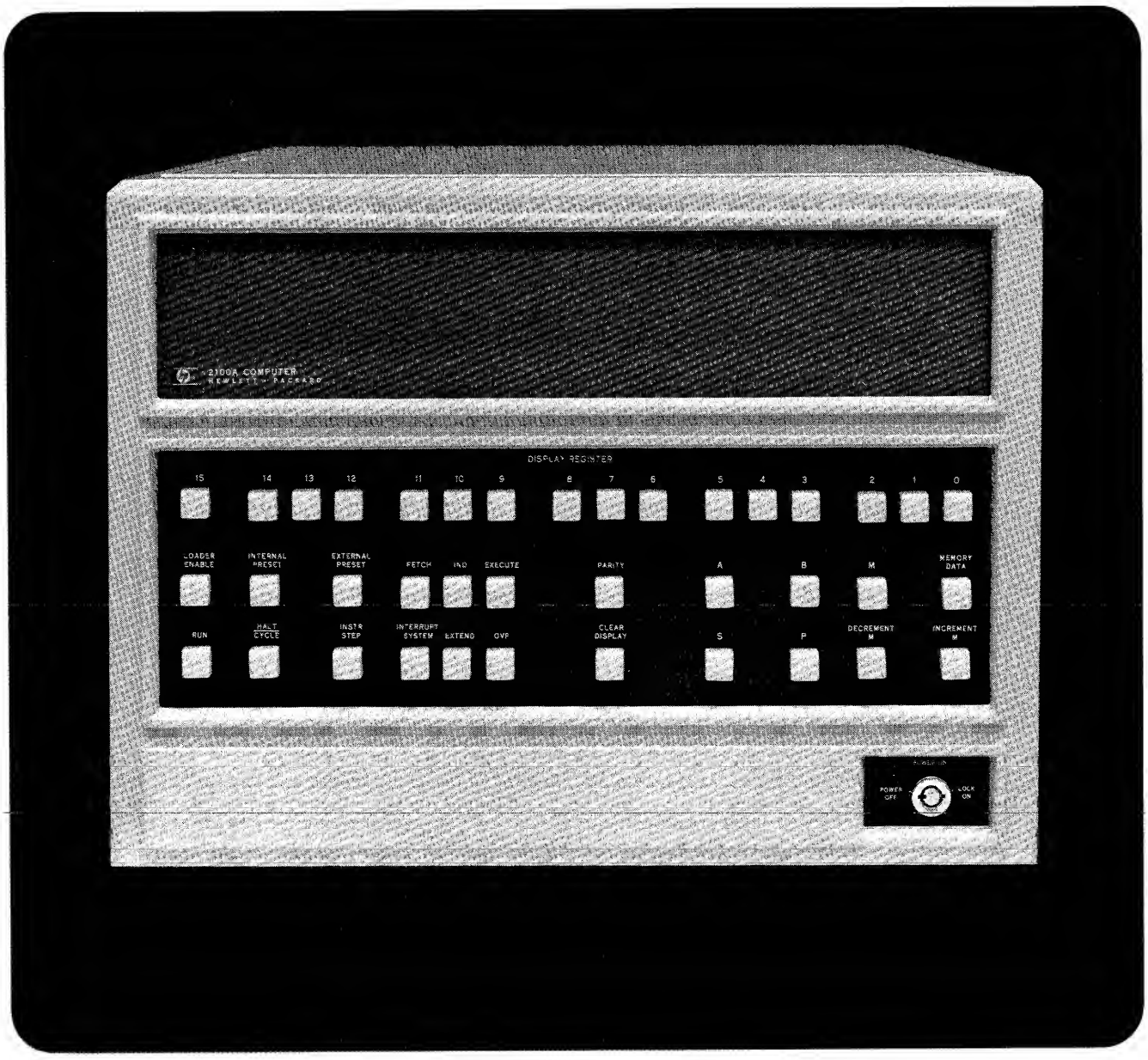


Figure 1. HP 2100A Computer

SYSTEM FEATURES

SECTION I

The Hewlett-Packard 2100A Computer is a compact data processor featuring a powerful, extended instruction set, plug-in interfaces, and modular software. Standard features include memory parity generation and checking, memory and I/O protect for executive systems, extended arithmetic capability, and power fail interrupt with automatic restart. Optional features include two-channel direct memory access, multiplexed input/output, a controller panel, and the I/O interfaces. The controller panel, which provides a minimum of controls and indicators, is available for applications where the full complement of controls and indicators provided on the operator panel is not necessary.

When integrated into one of the many possible system configurations the 2100A Computer, with its modular Hewlett-Packard software, provides a flexible, efficient systems package. The programming languages used are those most commonly encountered in data processing. Virtually all software written for the Hewlett-Packard 2114/2115/2116-series computers may be used with the 2100A Computer.

56-CHANNEL INPUT/OUTPUT

Interfacing of peripheral devices is accomplished by plug-in interface cards. The computer mainframe can accommodate up to 14 interface cards, expandable to a total of 45 when the optional 2155A I/O Extender is used. Interrupt and addressing capabilities are present for 56 channels so that, using multiplexed I/O and an external controller, up to 56 devices can be handled. Interface cards are available for a wide variety of peripheral devices, and virtually all interfaces used in 2114/2115/2116-series computers may be used with the 2100A Computer. No power supply extenders are necessary for any combination of interfaces installed.

All I/O channels are buffered and bi-directional, and are serviced through a multilevel priority interrupt structure. The two optional direct memory access (DMA) channels are program-assignable to any two of the 14 interface slots in the mainframe, expandable to 45 slots if DMA is also installed in the extender, and can be dynamically reassigned. DMA transfers occur on a cycle-stealing basis, not subject to the I/O priority structure. The total bandwidth through both DMA channels is more than one million words per second.

HEWLETT-PACKARD SOFTWARE

Software for the 2100A Computer includes four high-level programming languages: HP FORTRAN, HP FORTRAN IV, HP ALGOL, and HP BASIC, plus an efficient, extended assembler which is callable by FORTRAN and ALGOL. Utility software includes a debugging routine, a symbolic editor, and a library of commonly used computational procedures such as Boolean, trigonometric, and plotting functions, real/integer conversions, natural log, square root, etc.

Hewlett-Packard provides several systems built around BASIC interpreters. The single-terminal BASIC system allows the user to prepare and run BASIC language programs conversationally through a teleprinter. Programs can also be entered through a tape reader and punched out on tape punches. A memory of at least 8K words is required. A similar system, Educational BASIC, allows BASIC programs to be translated from marked cards. The time-shared BASIC systems provide an extended version of the BASIC language to 16 or 32 users simultaneously. The extensions to BASIC allow the user to store and access large amounts of data in an external mass memory, to manipulate strings of characters, and to store and retrieve programs in mass memory.

Several operating systems are available, covering a wide range of applications. The Basic Control System, which simplifies the control of input/output operations, also provides relocatable loading and linking of user programs. The time-shared systems, using conversational BASIC language, permit up to 32 terminals to be connected to the system, either directly or by telephone lines via Dataphones. The Hewlett-Packard Real-Time Executive (RTE) system, permits several programs to run in real-time concurrently with general purpose background programs. This allows multiple data processing capabilities where separate computers are not economically feasible. The user can write programs in HP Assembly, FORTRAN, or ALGOL languages. A Magnetic Tape System and a Disc Operating System are also available. These systems greatly increase the speed and simplicity of assembling, compiling, loading, and executing user programs.

SYSTEM EXPANSION FEATURES

Memory sizes for the 2100A Computer are available in six configurations: 4K, 8K, 12K, 16K, 24K and 32K. All core memory is accommodated in the computer mainframe and is field-installable.

Figure 2 illustrates the configuration of the basic 2100A Computer and the expansion capabilities of memory and input/output. This figure approximately represents the top view and layout of the computer. For 4K or 8K memory, a card with the appropriate stack configuration is installed in position A. For 12K or 16K memory, the appropriate combination of 4K and 8K stacks is installed in positions A and B. For 24K, positions A and B have 8K stacks and an 8K stack is added in position C. For 32K, a final 8K stack is added in position D.

Expansion of input/output beyond the capability of the mainframe is accomplished by plugging an extender interface card into the highest address I/O slot (represented by E in figure 2), in place of an I/O interface card. This card is then cabled to an equivalent card in the 2155A I/O Extender Unit. The address formerly assigned to slot E, and all higher addresses, will be available in the extender unit.

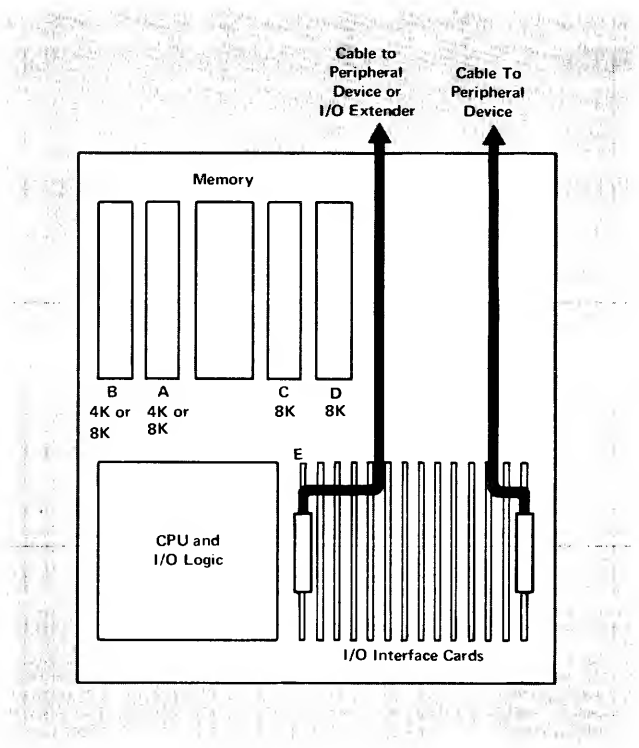


Figure 2. Internal Configuration

MULTIPROGRAMMING SYSTEMS

Multiprogramming systems allow several users at various terminals to utilize the same computer on an apparently simultaneous basis. In such an environment the computer must respond quickly to external events, and have sufficient capacity to handle a wide variety of programs at varying speeds. The 2100A Computer, equipped with suitable peripherals and executive software, is capable of meeting these demands while providing truly user-oriented operating features. Two Hewlett-Packard systems provide multiprogramming capability: the Time-Shared BASIC system and the Real-Time Executive (RTE) system.

TIME-SHARED BASIC SYSTEM

The time-shared system combines standard, low-cost computer hardware with BASIC language programming to provide simultaneous servicing of either 16 or 32 users. In addition, HP conversational BASIC is an easy-to-learn yet powerful language. The extended features of HP Time-Shared BASIC cover a wide range of capabilities from simple arithmetic operations to advanced programming projects requiring the use of matrices, strings, and files. Special applications packages include computer-assisted instruction and a business and finance library. Also, for users requiring a general-purpose single-terminal capability, the system can be used as a dedicated system which offers the flexibility of HP FORTRAN, ALGOL and Assembly language programming.

Also of major importance, every HP Time-Shared System is supplied completely operational, ready for connection to the user's terminals. A minimum system consists of a 2100A Computer, mass storage unit, high-speed tape reader (photoreader), two-bay cabinet, and a heavy-duty system teleprinter. Terminals can be connected through standard telephone communication equipment or cabled directly (distances up to one mile), resulting in even greater savings.

Each of the system's users receives equivalent servicing (depending on how many terminals are currently in use). The programmer is not placed in the position of having to constantly bid for the computer's attention. Response time to program statement editing is typically less than 0.3 second. No valuable time is lost waiting for an input terminal or in getting through the batch-processing loop. Users also appreciate the fully conversational language. Programs are checked line-by-line during entry, eliminating the need for repeated cycles through the program. The use of ordinary English for communication simplifies the task of learning and maintaining programming efficiency. The simplicity of BASIC permits the inexperienced user to perform useful programming after only a few hours of training or self-teaching.

Mass storage provides, in its minimum configuration, 400,000 characters of file storage, equivalent to 200 programs of 100 statements each. Data security is provided by means of unique user identification codes.

Several software packages are available to enhance system usefulness for certain applications. One such package is the HP Computer-Assisted Instruction (CAI) program which provides mathematics drill and practice for grades 1 through 6. The mathematics instruction program is written in BASIC language and the system treats it as a library program. Thus a system can be used concurrently for CAI and general-purpose problem solving. The Hewlett-Packard CAI curriculum is based on the Stanford Project Drill and Practice Program in Fundamentals of Arithmetic.

GENERAL SPECIFICATIONS

BASIC CHARACTERISTICS

- 16-bit word length; 17th bit for memory parity checking
- Parallel logic
- Power fail interrupt, with automatic restart
- Rack-mountable

MEMORY

- Magnetic core storage
- 980 nanosecond cycle time
- Parity generation and checking is standard in all units
- Six memory sizes available, 4,096 to 32,768 words; field-expandable by plug-in cards
- 1024-word page size
- Protected 64-word block for stored loader

PROCESSOR

- 80 basic instructions, including extended arithmetic
- Up to eight instructions may be combined into one word (register reference group)
- Two accumulators, addressable as memory locations
- Unlimited levels of indirect addressing allowed
- Six working registers, may be selected for display and instant modification (A, B, T, P, M, S)
- Illuminated control pushbuttons allow simultaneous display and control of internal functions
- All instructions fully executed in 1.96 microseconds, except ISZ and extended arithmetic (2.94 to 16.7 microseconds)
- Only 980 nanoseconds added for each level of indirect addressing
- Memory and I/O protection is standard

SOFTWARE

- FORTRAN, FORTRAN IV, ALGOL, and BASIC languages
- Extended Assembly language
- Editor, subroutine library, Formatter, and Debug routine
- Several operating systems, including
  - Basic Control System
  - Magnetic Tape System
  - Disc Operating System
  - Time-Shared BASIC System

INPUT/OUTPUT SYSTEM

- 14 internal I/O channels, externally expandable to 45
- Optional multiplexed I/O extends capacity to 56 channels; may be plugged into any slot
- All channels buffered and bi-directional
- Multilevel priority interrupt for device servicing
- Peripherals interfaced simply with plug-in cards
- Optional dual-channel direct memory access, can transfer 1,020,400 words per second
- General-purpose interface cards available

PERIPHERALS

- Magnetic Tape
  Read and write 9-track IBM-compatible magnetic tape, 800 BPI, at speeds of 25, 37.5, or 45 inches per second; also read and write 7-track IBM-compatible magnetic tape at speeds of 25, 37.5, or 45 inches per second with switch-selectable densities of 200, 556, and 800 BPI
- Disc/Drum Memory
  Fixed head-per-track design for rapid access, capacities range from 262,144 to 1,048,576 words
- Cartridge Disc
  Moving-head disc for low-cost mass storage; capacities from 1.2 million to 4.8 million words
- Disc File
  Moving-head mass storage; 11.7 million words per drive, 8 drives maximum
- Card Reader
  Reads punched 80-column cards, 12 bits in parallel, at 1000 cards per minute
- Mark Reader
  Reads punched and pencil-marked cards at 200 cards per minute
- Line Printers
  Print 120 or 132 columns per line at 300 lines per minute; ASCII 64-character set. Also from 356 lines per minute (80 columns) to 1110 lines per minute (20 columns); 64-character set
- Keyboard Display Terminal
  CRT screen displays 25 lines, 72 characters per line; standard teleprinter keyboard plus 10-key numerical keyboard; speeds of 10 to 200 characters per second, switch selectable
- Tape Readers
  Read 5- and 8-level punched paper tape at up to 500 characters per second; with or without automatic reroller
- Tape Punch
  Punch 5- and 8-level code at 120 characters per second; also 5- and 8-level code at 75 characters per second
- Teleprinter
  Keyboard with punch, read, and print capabilities; 10 characters per second; ASCII character set
- Data Communications
  Asynchronous data set interface, 75 to 2400 bits per second, full- and half-duplex; also synchronous receive and transmit data set interfaces, 1200 to 9600 bits per second; automatic dialing capability available

Hewlett-Packard also offers a General Business and Finance Library (available on a subscription basis) for use on its time-share systems. This means the end-user who installs a time-share system can expect it to be used by all departments, including accounting, engineering, production, finance, inventory control and so on. Library programs are included for general ledger accounting, depreciation analysis, cash investment analysis, and file handling.

REAL-TIME EXECUTIVE

The Real-Time Executive (RTE) schedules the running of programs on a priority basis under both time and event control. It supervises all interrupts, I/O operations, and bulk memory transfers plus the housekeeping necessary to a system where multiprogramming is performed. The availability of background programming provides maximum utilization of hardware and a practical solution where separate computers are not economically feasible. Time not needed for real-time processing may be used for computation, compiling, program debugging and other operations while keeping the foreground and executive safe through memory protection of both core and mass storage.

Programs for the system can be written using HP Assembly, HP FORTRAN, HP FORTRAN IV, or HP ALGOL languages. The number of programs in the system is limited only by mass storage capacity and the dynamics of the application. Core memory requirements are minimized by configuring a tailored-system from standard modules. The Real-Time Executive is configured and operated on the same computer system. A minimum of 16K memory is required.

OPERATING SYSTEMS

Hewlett-Packard operating systems built around the 2100A Computer simplify user I/O operations and the compilation and execution of user programs. Standard operating systems include the Basic Control System (BCS), the Magnetic Tape System (MTS), Disc Operating System (DOS), and the Data Acquisition and Control Executive (DACE).

BASIC CONTROL SYSTEM

The Basic Control System provides an efficient loading and input/output control capability for relocatable programs produced by the HP Assembler or HP FORTRAN and HP ALGOL compilers. The Basic Control System performs the following functions: loads, links, and initiates the execution of relocatable object programs and library subroutines; produces complete absolute binary output tapes for "production" programs; processes all input/output operations, using software buffering, simple calling sequences, logical I/O unit numbers, and modular input/output drivers which perform the actual I/O operations on all peripheral devices.

Two routines associated with the Basic Control System, though not physically part of it, are Prepare Control System, which establishes the Basic Control System configuration, and Debugging System, which is loaded with a user program to aid in program testing. The minimum memory requirement is 4K.

MAGNETIC TAPE SYSTEM

The Magnetic Tape System uses magnetic tape to provide vastly increased assembly, compilation, and loading speed. The system can handle both absolute and relocatable programs from FORTRAN, ALGOL, BASIC, Assembly language through the Symbolic Editor and Basic Control System software. A batch processing mode is included. The Magnetic Tape System is created by transferring software programs from paper tape to magnetic tape. In the magnetic tape environment, programs are loaded into core automatically by a supervisory program that operates in response to user requests. Once the Magnetic Tape System is configured, the magnetic tape is organized into program files (for absolute programs and linkable subroutines) and the remainder of the magnetic tape is available to executing programs for storage of temporary data and scratch use. A minimum of 8K memory is required.

DISC OPERATING SYSTEM

The Disc Operating System (DOS) speeds and simplifies the creation, checkout and execution of programs. With this system a user can assemble or compile, load, and execute programs under control of the teleprinter keyboard or batch input device. Executive software automatically moves the assembler, compiler, loader, source programs, and intermediate versions of the program between bulk storage and core memory, as required, without the handling of paper tapes at each step. Very large programs can be implemented through program segmentation and overlays.

For batch processing, DOS operates continuously to do a series of jobs, each of which can have independent programs and different I/O configurations. Source files can be edited, and a variety of files can be stored, dumped, and used as input to programs.

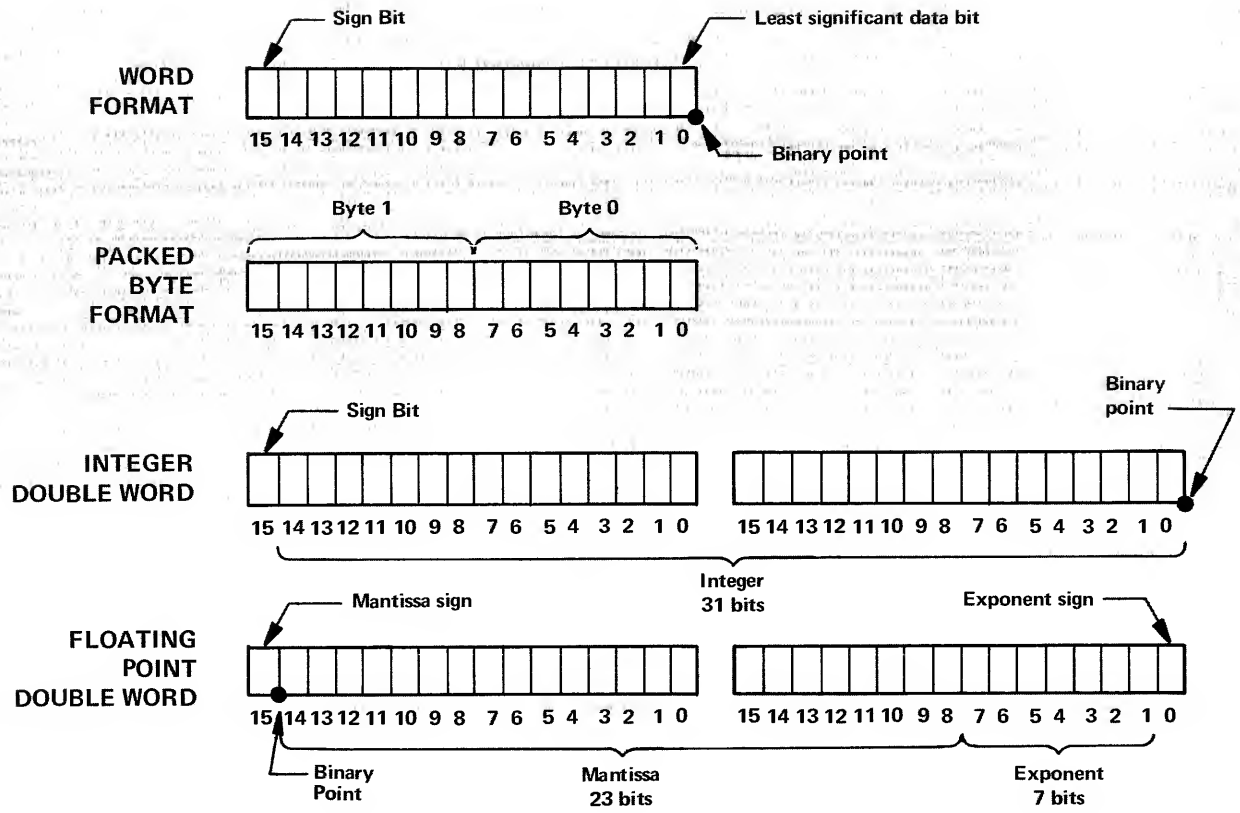
The Disc Operating System comprises a series of relocatable binary software modules. Because each module is an independent, general purpose program, the configuration of DOS is very flexible. A separate absolute program accepts the software modules and generates a configured DOS following dialogue-type instructions from the user. Certain DOS modules may be resident in either core or bulk storage. In a minimum 8K core memory system, all possible modules are resident in bulk storage; but a 16K or larger memory allows more modules to be core resident for faster processing. Two versions of DOS are available: one for fixed-head discs and drums, and one for moving-head discs.

An absolute copy of the configured DOS is retained in bulk storage and is hardware protected against alteration.

DATA ACQUISITION AND CONTROL EXECUTIVE

The Data Acquisition and Control Executive (DACE) is a general-purpose operating system for scheduling data acquisition and control functions. Such functions typically include taking measurements, computing answers to problems, logging and displaying the results, and providing outputs for controlling equipment and processes. In application, these functions are written as "tasks" for the DACE system; DACE then schedules execution of the tasks in real time, at intervals that can be specified in hours, minutes, and seconds. Since the scheduled execution of tasks is automatic, the user does not need to be concerned with operational details while the system is taking data or performing control functions. Full attention can be given to evaluating the results of system operation.

Tasks may be written in HP FORTRAN or Assembly language, and can refer to ALGOL procedures. Because tasks are typically short and uncomplicated, debugging is relatively easy. During operation, DACE provides for switching the system to a manual mode in which task constants can be examined and modified conversationally, using the system teleprinter. In addition, operator intervention is also possible in automatic mode, for such purposes as selective activation and deactivation of specific tasks or selection of alternative actions within tasks. DACE requires a minimum of 8K memory.



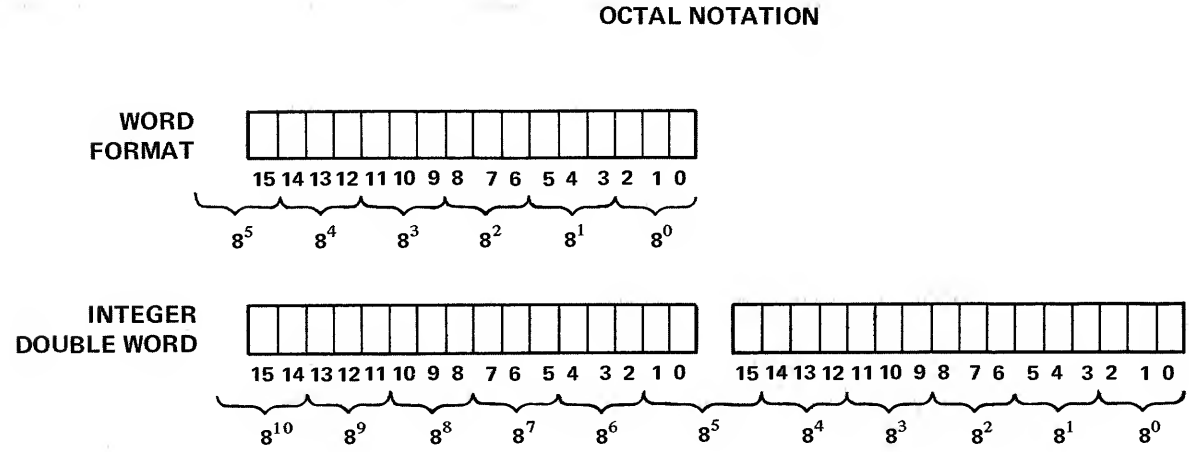


Figure 3. Data Formats and Octal Notation

DATA FORMATS

The basic data format for the 2100A Computer is a 16-bit word. Bit positions are numbered from 0 through 15, in order of increasing significance. Bit position 15 of the data format is used for the sign bit; a "0" in this position indicates a positive number and a "1" indicates a negative number. The data is assumed to be a whole number, thus the binary point is assumed to be to the right of the number.

The basic word, as shown in figure 3, can also be divided into two 8-bit bytes or combined to form a 32-bit doubleword. The byte format is used for character-oriented input/output devices. Packing of the two bytes into one word is accomplished by the software drivers. In I/O operations the higher order byte (Byte 1) is the first to be transferred.

The integer doubleword format is used for extended precision arithmetic in conjunction with the ten extended arithmetic instructions. Bit 15 of the most significant word is the sign bit, and the binary point is assumed to be to the right of the least significant word. The integer value is expressed by the remaining 31 bits. When addressing a doubleword in memory, the address refers to the least significant word location; the next higher location contains the most significant word. When loaded into the accumulators, the B-register contains the most significant word and the A-register contains the least significant word.

The floating point doubleword format is used with floating point software. Bit 15 of the most significant word is the mantissa sign bit and bit 0 of the least significant word is the exponent sign bit. Bits 1 through 7 are used to express the exponent, and the remaining bits (8 through 15 of the least significant word and 0 through 14 of the most significant word) are used to express the mantissa. The mantissa is assumed to be a fractional value, thus the binary point appears to the left of the mantissa. Software converts decimal numbers to this binary form and normalizes the quantity expressed (sign and leading mantissa bit differ). If either the mantissa or the exponent is negative, that part is stored in two's complement form. The number must be in the approximate range of 10^{-38} through 10^{+38}.

Figure 3 also illustrates the octal notation of data for both single-length and double-length words. Each group of three bits, beginning at the right, is combined to form an octal digit. Each digit to the left increases in significance. A single-length 16-bit word can therefore be fully expressed by six octal digits and a double-length 32-bit word can be fully expressed by 11 octal digits. Octal notation is not shown for byte or floating point formats, since bytes normally represent characters and floating point numbers are given in decimal form.

For single-word data, the range of representable numbers is +32,767 to -32,768 (decimal), or +77,777 to -100,000 (octal). For doubleword integer data, the range is +2,147,483,647 to -2,147,483,648 (decimal), or +17,777,777,777 to -20,000,000,000 (octal).

MEMORY ADDRESSING

The 2100A Computer can be equipped with any one of six memory configurations, from 4K to 32K. (K = 1024 words.) The available configurations, which determine the addressing range, are: 4K, 8K, 12K, 16K, 24K, and 32K.

PAGING

The computer memory is logically divided into pages of 1024 words each. A page is defined as the largest block of memory which can be directly addressed by the memory address bits of a memory reference instruction (single-length). These memory reference instructions have 10 bits to specify a memory address, and thus the page size is 1024 locations (2000 in octal notation). Octal addresses for each page, up to the maximum memory size, are given in table 1.

Provision is made to address directly one of two pages: page zero (the base page, consisting of locations 00000 through 01777), and the current page (the page in which the instruction itself is located). Memory reference instructions include a bit (bit 10) reserved to specify one or the other of these two pages. To address locations in any other page, indirect addressing is used. Page references are specified by bit 10 as follows:

Logic 0 = Page Zero (Z)
Logic 1 = Current Page (C)

INDIRECT ADDRESSING

All memory reference instructions reserve a bit to specify direct or indirect addressing. For single-length memory reference instructions, bit 15 of the instruction word is used; for extended arithmetic memory reference instruc-

Table 1. Memory Pages

MEMORY SIZE	PAGE	OCTAL ADDRESSES
	0	00000 to 01777
	1	02000 to 03777
	2	04000 to 05777
4K ↓	3	06000 to 07777
	4	10000 to 11777
	5	12000 to 13777
	6	14000 to 15777
8K ↓	7	16000 to 17777
	8	20000 to 21777
	9	22000 to 23777
	10	24000 to 25777
12K ↓	11	26000 to 27777
	12	30000 to 31777
	13	32000 to 33777
	14	34000 to 35777
16K ↓	15	36000 to 37777
	16	40000 to 41777
	17	42000 to 43777
	18	44000 to 45777
	19	46000 to 47777
	20	50000 to 51777
	21	52000 to 53777
	22	54000 to 55777
24K ↓	23	56000 to 57777
	24	60000 to 61777
	25	62000 to 63777
	26	64000 to 65777
	27	66000 to 67777
	28	70000 to 71777
	29	72000 to 73777
	30	74000 to 75777
32K ↓	31	76000 to 77777

tions, bit 15 of the address word is used. Indirect addressing uses the address part of the instruction to access another word in memory, which is taken as a new memory reference for the same instruction. This new address word is a full 16 bits long, 15 bits of address plus another direct/indirect bit. The 15-bit length of the address permits access to any location in memory. If bit 15 again specifies indirect addressing, still another address is obtained; this multiple-step indirect addressing may be done to any number of levels. The first address obtained in the indirect phase which does not specify another indirect level becomes the effective address for the instruction. Direct or indirect addressing is specified by bit 15 as follows:

Logic 0 = Direct
Logic 1 = Indirect

RESERVED LOCATIONS

The first 64 memory locations of the base page (octal addresses 00000 through 00077) are reserved as listed below. The first two addresses are the A and B flip-flop register addresses and are not considered as core storage locations. (The actual corresponding core locations can, however, be loaded and read via the operator panel.) Locations 4 through 77 are reserved in the sense that interrupt wiring is present for the priority order given. As long as the locations do not have actual interrupt assignments (as determined by the input/output devices included in the user's system), these locations may be used for program purposes.

00000	Address of A-register.
00001	Address of B-register.
00002 00003	For exit sequence if A and B contents are used as executable words.
00004	Interrupt location, highest priority (reserved for power fail interrupt).
00005	Reserved for memory parity and memory protect interrupts.
00006	Reserved for direct memory access.
00007	Reserved for direct memory access.
00010 thru 00077	Interrupt locations in decreasing order of priority.

The last 64 locations of memory (any size) are reserved for the basic binary loader. The basic binary loader is a permanently resident program to permit loading of binary information from punched paper tape (or disc, etc.) into memory. Unless specifically enabled by a panel switch, the loader locations are protected so they may not be altered or used in any way.

NONEXISTENT MEMORY

Nonexistent memory is defined as those memory locations not physically implemented in the machine (up to the maximum of 32K) and the last 64 locations of implemented memory when not enabled from the front panel. Any attempt to write into nonexistent memory will be ignored (no operation). Any attempt to read from a nonexistent memory location will return an all-zero word; no parity error occurs.

HARDWARE REGISTERS

The 2100A Computer has six 16-bit working registers, two one-bit registers, and (on the operator panel) one 16-bit

display register. The functions of these registers are described as follows.

M-REGISTER. The M-register holds the address of the memory cell currently being read from or written into.

T-REGISTER (MEMORY DATA). All data transferred into or out of memory is routed through the memory data register. When displayed, the display indicates the contents of the memory location currently pointed to by the M-register. The displayed data will go back into that location when any other action is taken (such as displaying some other register or beginning a run operation).

P-REGISTER. The P-register holds the address of the next instruction to be fetched out of memory. Since this is a "look-ahead" register, the P-register value will frequently differ from the M-register value. Table 2 lists P- and M-register contents for each of five different computer states, assuming the computer is halted.

A-REGISTER. The A-register is an accumulator, holding the results of arithmetic and logical operations performed by programmed instructions. This register may be addressed by any memory reference instruction as location 00000, thus permitting inter-register operations such as "add B to A", "compare B with A", etc., using a single-word instruction.

B-REGISTER. The B-register is a second accumulator, which can hold the results of arithmetic operations completely independent of the A-register. The B-register may be addressed by any memory reference instruction as location 00001 for inter-register operations with A.

S-REGISTER. The switch (S) register is a 16-bit utility register. In the halt mode, it may be manually loaded via the display register. In the run mode it may be addressed as an I/O device (select code 01) and receive and read back data to and from the accumulators.

EXTEND. The extend bit (E) is a one-bit register, and is used to link the A- and B-registers by rotate instructions or to indicate a carry from bit 15 of the A- or B-registers by an add instruction (ADA, ADB) or increment instruction (INA or INB, but not ISZ) which references these registers. This is of significance primarily for multiple-precision arithmetic. If already set, the extend bit is not complemented by a carry. It may be set, cleared, complemented, or tested by program instruction. The extend bit is set when the EXTEND light is on ("1") and clear when off ("0").

OVERFLOW. The overflow bit is a one-bit register which indicates that an add instruction (ADA, ADB), divide instruction (DIV), or an increment instruction (INA or INB, but not ISZ) referencing the A- and B-registers has caused (or will cause) the accumulators to exceed the maximum positive or negative number which they can contain. By program instructions, the overflow bit may be cleared, set, or tested. The OVF light remains on until the bit is cleared by an instruction and is not complemented if a second overflow occurs before being cleared. It will not be set by any shift or rotate instructions, except ASL (refer to definition in section III).

Table 2. P- and M-Register Indications

COMPUTER STATUS	P-REGISTER contains address of	M-REGISTER contains address of
FETCH	Current instruction	Last memory access
INDIRECT (after FETCH)	Current instruction	Current instruction
INDIRECT (after INDIRECT)	Current instruction	Last memory access
EXECUTE (after FETCH)	Next instruction	Current instruction
EXECUTE (after INDIRECT)	Next instruction	Last memory access

DISPLAY REGISTER. The display register is included on the standard operator panel. It provides a means for displaying and modifying the contents of any of the six 16-bit working registers when the computer is in the halt mode. Each pushbutton is illuminated to indicate a content of "1", and is non-illuminated to indicate a content of "0". Each time a pushbutton is pressed, the content changes state. When the computer is in the run mode, the display register permanently displays the S-register contents.

INSTRUCTION FORMATS

Instructions for the 2100A Computer are classified according to format. The five formats used are illustrated in figure 4 and are described as follows. In all cases where a single bit is used to select one of two cases (e.g., D/I), the choice is made by coding a logic 0 or 1 respectively (i.e., 0/1).

MEMORY REFERENCE. This class of instructions combines an instruction code and a memory address into one word. This type of instruction is therefore used to execute some function involving data in a specific memory location. Examples are storing, retrieving, and combining memory data to or from the accumulators, or causing the program to jump to the specified location.

The cell referenced (i.e., the absolute address) is determined by a combination of the ten memory address bits in the instruction word (0 through 9) and five bits (10 through 14) assumed from the current condition of the P-register. This means that memory reference instructions can directly address any word in the current page; additionally, if the instruction is given in some location other than the base page (page zero), bit 10 of the instruction word doubles the addressing range to 2048 words by allowing selection of either page zero or current page. (This causes bits 10 through 14 of the address in the M-register to be reset to zero, instead of assuming the current indication of the P-register.) This feature provides a convenient linkage between all pages of memory, since page zero can be reached directly from any other page.

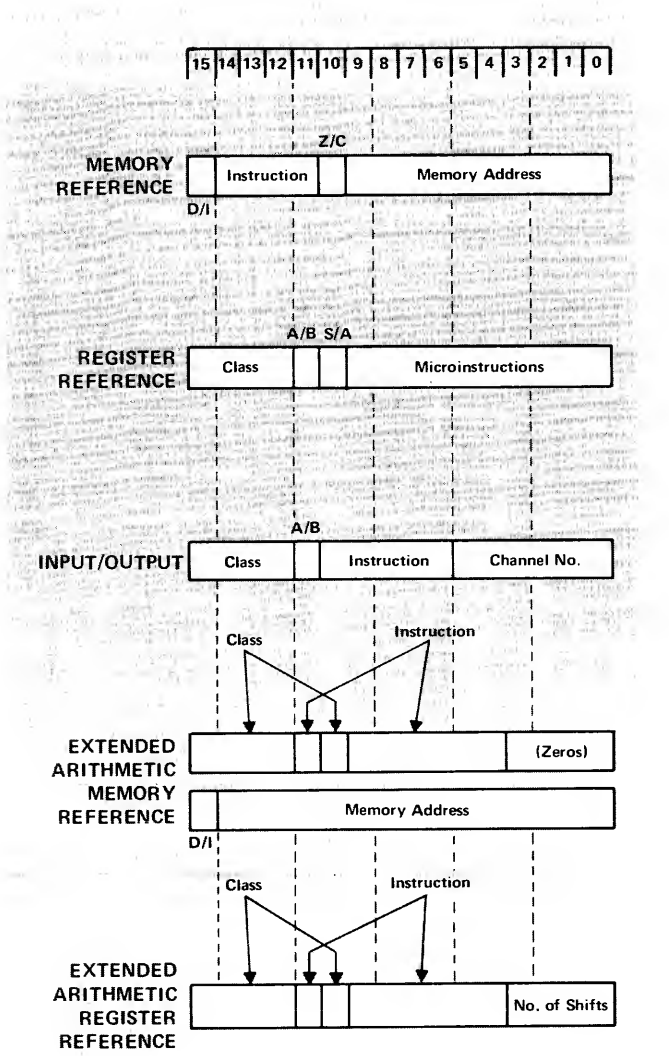


Figure 4. Instruction Formats

As discussed earlier, bit 15 is used to specify direct or indirect addressing. Also note that since the A- and B-registers can be addressed, any single-word memory reference instruction can apply to either of these registers as well as to memory cells. For example, ADA 0001 means add the contents of the B-register (its address being 0001) to the A-register; specify page zero for these operations, since the A and B register addresses are on page zero.

REGISTER REFERENCE. These instructions, in general, manipulate bits in the A-, B-, and E-registers. There is no reference to memory. This type includes 39 basic instructions, which are combinable to form a one-word multiple instruction that can operate in various ways on the contents of the A-, B-, or E-registers. The 39 instructions are divided into two subgroups, the shift-rotate group (SRG) and the alter-skip group (ASG). These subgroups are specified by bit 10. Typical operations are clear and/or complement a register, conditional skips, and register increment.

INPUT/OUTPUT. The input/output class of instructions uses bits 6 through 11 for a variety of I/O instructions, and bits 0 through 5 to apply the instruction to a specific I/O channel. This provides a means of controlling all devices connected to the I/O channels, and for transferring data in or out. Also included in this group are instructions to control the interrupt system, overflow bit, and computer halt.

EXTENDED ARITHMETIC MEMORY REFERENCE. Like the single-word memory reference instruction above, the complete instruction includes an instruction code and a memory address. In this case, however, two words are required. The first word specifies the extended arithmetic class (bits 12 through 15 and 10) and the instruction code (bits 4 through 9 and 11). Bits 0 through 3 are not needed and are coded with zeros. The second word specifies the memory address of the operand. Since a full 15 bits are used for the address, this type of instruction may directly address any location in memory. As with all memory reference instructions, bit 15 may be used to specify indirect addressing. Operations provided by this class of instructions are integer multiply and divide (using double-length product and dividend), and double load and double store.

EXTENDED ARITHMETIC REGISTER REFERENCE. This class of instructions provides long shifts and rotates on the combined A- and B-registers. Bits 12 through 15 and 10 identify the extended arithmetic class, and bits 4 through 9 and 11 specify the direction and type of shift. Bits 0 through 3 are used to specify the number of shifts, which can range from 1 to 16 places.

INTERRUPT SYSTEM

The computer interrupt system has 60 distinct interrupt levels. Each level has a unique priority assigned to it, and is associated with a numerically corresponding interrupt location in core memory.

As an example of the simplicity of this system: a service request from I/O channel 13 will cause an interrupt to core location 00013. The request for service will be granted on a priority basis higher than channel 14 but lower than channel 12. Thus a transfer in progress via channel 14 would be suspended to let channel 13 proceed, but a transfer via channel 12 could not be interrupted by channel 13.

Under program control, any device may be selectively enabled or disabled, thus switching the device in or out of the interrupt structure. In addition the entire interrupt system may be enabled or disabled under program control using a single instruction (excepting power fail and parity error interrupts).

Of the 60 interrupt levels, the two highest priority levels are reserved for hardware faults (power fail and parity error), the next two are reserved for DMA completion interrupts,

and the remaining 56 are available for the I/O device channels. Table 3 lists interrupt levels in order of priority. Note that interrupt facilities for I/O channels above 25 (octal) are available through use of an I/O extender or multiplexer.

Interrupt requests received while the computer is in halt mode will be processed, in order of priority, when the computer is put into run mode or is stepped single cycle.

Table 3. Interrupt Assignments

CHANNEL (Octal)	INTERRUPT LOCATION	ASSIGNMENT
04	00004	Power Fail Interrupt
05	00005	Memory Parity/Protect Interrupt
06	00006	DMA Channel 1 Completion Interrupt
07	00007	DMA Channel 2 Completion Interrupt
10	00010	I/O Device, highest priority
thru 25	00025	I/O Device (Mainframe)
thru 65	00065	I/O Device (Extender)
thru 77	00077	I/O Device (Multiplexer)

POWER FAIL INTERRUPT

The computer is equipped with power sensing circuits. When primary power to the computer fails or drops below a safe operating level while the computer is running, an interrupt to memory location 00004 is automatically generated. This interrupt is given the highest priority in the system, and cannot be turned off or disabled. Location 00004 is intended to contain a jump-to-subroutine instruction referencing the entry point of a shut-down program, but it may alternatively contain a HLT instruction. Interrupt capability for lower-priority functions is automatically inhibited while a power fail routine is in progress. Sufficient time is available between the detection of power failure and the loss of usable internal power to execute about 100 instructions. The shut-down program should be written to save the current state of the computer system in memory, and then must halt the computer. A sample program is given in table 4.

Since the restoration of power might be unattended by an operator, the user is given a switch-selectable option of what action the computer should take. With the switch set to the halt position, the computer will halt when power is restored, whether the computer was running or halted when the failure occurred. (No panel indication is given.) With the switch in the restart position, the automatic restart feature is enabled. After a built-in delay of about a second following return to normal power levels, another interrupt is generated, again to location 00004. This time the shut-down portion of the subroutine is skipped (see sample subroutine) and the power-up portion begins. If the computer was not running when the power failure occurred, the computer is halted. If the computer was running, the system conditions are restored and the computer continues operation from the point of interruption. Alternatively, if location 00004 contains a HLT instead of a jump to a subroutine, the computer will halt at this time and EXTERNAL PRESET (or PRESET on the controller panel) will light.

To allow for the possibility of a second power failure occurring while the power-up routine is in progress, the user should limit the combined total of instructions (for both shut-down and power-up) to less than 100. If the computer memory does not contain a subroutine to service the interrupt, location 00004 should contain a HLT 04 instruction (octal 102004).

A set control command (STC 04) must be given at the end of any restart routine. This command re-initializes the power fail logic and restores interrupt capability to lower priority functions. The EXTERNAL PRESET switch, when pressed, issues a similar command.

PARITY ERROR INTERRUPT

Parity checking of memory is a standard feature of the 2100A Computer. The parity logic continuously generates correct parity for all words written into memory and monitors the parity of all words read out of memory. Correct parity is defined as having the total number of "1" bits in a 17-bit memory word equal an odd value. If a "1" bit (or any odd number of "1" bits) is either dropped or added in the transfer process, a parity error signal is generated when the word is read out. Unless the error logic is specifically disabled by a CLF 05 instruction, the error signal causes an interrupt to location 00005.

Optionally (switch-selectable) the error signal may cause a halt, rather than an interrupt. The lighting of the HALT and PARITY indicators signal the fact that the halt was caused by a parity error. The PARITY light stays on until INTERNAL PRESET is pressed.

Assuming that the interrupt option is selected, the interrupt to location 00005 directs the computer to the entry point of a parity error subroutine. It is the user's decision as to what to do about a parity error; for example he may want to record the address of the error location, or abort a critical operation. In any case, the PARITY light is turned off as soon as the interrupt is acknowledged and normal operation may be resumed on exit from the subroutine. A STF 05 instruction should be given at the end of the subroutine to re-initialize the logic.

In conjunction with the memory protect feature, it is possible to determine the address of the error location.

Table 4. Sample Power Fail Subroutine

LABEL	OPCODE	OPERAND	COMMENTS
PFAR	NOP		Power Fail/Auto Restart Subroutine
	SFC	4B	Skip if interrupt was caused by a power failure
	JMP	UP	Power is being restored, reset state of computer system
DOWN	STA	SAVA	Save A-register contents
	CCA		Set switch indicating that the computer was running when
	STA	SAVR	power failed
	STB	SAVB	Save B-register contents
	ERA,ALS		Transfer E-register content to A-register bit 15
	SOC		Increment A-register if Overflow
	INA		is set
	STA	SAVEO	Save E- and O-register contents
	LDA	PFAR	Save contents of P-register at time of
	STA	SAVP	power failure
	LIA	1B	Save contents of
	STA	SAVS	S-register
	⋮		Insert user-written routine to save I/O device states
	CLC	4B	Turn on restart logic so computer will restart when power is restored after momentary power failure
	HLT		Shutdown
UP	LDA	SAVR	Was computer running when
	SZA,RSS		power failed?
	JMP	HALT	No
	CLA		Yes, reset computer Run switch to
	STA	SAVR	initial state
	LDA	FENCE	Restore the memory protect
	OTA	5B	fence register contents
	⋮		Insert user-written routine to restore I/O device states
	LDA	SAVEO	Restore the contents
	CLO		of the
	SLA,ELA		E-register and
	STF	1B	O-register
	LDA	SAVS	Restore the contents of the
	OTA	1B	S-register
	LDA	SAVA	Restore A-register contents
	LDB	SAVB	Restore B-register contents
	STC	4B	Reset power fail logic for next power failure
	STC	5B	Turn on memory protect
	JMP	SAVP,I	Transfer control to program in execution at time of power failure
HALT	HLT		Return computer to halt mode
FENCE	OCT	2000B	Fence address storage (must be updated each time fence is changed)
SAVEO	OCT	0	Storage for E and O
SAVA	OCT	0	Storage for A
SAVB	OCT	0	Storage for B
SAVS	OCT	0	Storage for S
SAVP	OCT	0	Storage for P
SAVR	OCT	0	Storage for Run switch

The error address will automatically be loaded into the violation register of the memory protect logic, and from there it is accessible to the programmer. (See following discussion of memory protect interrupt.)

It is recommended, on discovery of a parity error, that the entire program or set of data containing the error location be reloaded. However, knowing the address and contents of the error location, the user may be able to determine what operations have taken place as a result of reading the erroneous word. For example, if the word was an instruction, several other locations may be affected. By individually checking and correcting the contents of all affected locations, the user may resume running his program without a complete reload. If software is being generated, this may also need to be corrected.

MEMORY PROTECT INTERRUPT

Memory protect for the 2100A Computer is a standard feature. With this capability a selected block of memory of any size, from a settable fence address downward, is protected against alteration by memory reference instructions (excluding A- and B-register addresses, which may be freely addressed by any memory reference instruction except JMP). Also, when enabled, it prohibits the execution of all I/O instructions except those referencing I/O address 01 switch and overflow registers. This second feature limits the control of input/output operations to interrupt control only. Then, by programming the system to direct all I/O interrupts to an executive program in protected memory, the executive program can have exclusive control of the I/O system.

The memory protect logic is disabled by any interrupt (except if the interrupt location contains an input/output group instruction) and is re-enabled by a STC 05 instruction at the end of each interrupt subroutine. In the halt mode, memory protect is also disabled by the INTERNAL PRESET switch.

Programming rules pertaining to the use of memory protect, assuming the logic is enabled, are as follows:

a. Location 00002 is the lower boundary of protected memory. (Locations 00000 and 00001 are the A- and B-register addresses.)

b. JMP instructions may not reference the A- or B-registers. JSB, however, may do so.

c. The upper boundary is loaded into the fence register from the A- or B-register by an OTA or OTB instruction with select code 05. Memory locations below (but not including) this address are protected.

d. Execution will be inhibited and an interrupt to location 00005 will occur if a JMP, JSB, ISZ, STA, STB, or DST instruction directly or indirectly addresses a location in protected memory, or if any I/O instruction is attempted (including halt, but excluding those addressing select code 01, the S and overflow registers).

e. Any instruction not mentioned in "d" is legal, even if it does reference protected memory. In addition, indirect addressing through protected memory by those memory reference instructions listed in "d" is legal, provided the final effective address is outside protected memory.

Following a memory protect interrupt, the address of the illegal instruction will be present in the violation register. This address is made accessible to the programmer by a LIA 05 or LIB 05 instruction, which loads the address into the A- or B-register.

Since parity error and memory protect share the same interrupt location, it is necessary to distinguish which type of error is responsible for the interrupt. If after the LIA/B 05 instruction (preceding paragraph), bit 15 of the A-/B-register is a "1", parity error is indicated; if bit 15 is a "0", memory protect violation is indicated. In either case, the remaining bits of the register give the address of the error location.

Table 5 illustrates a sample memory protect and parity error subroutine. An assumption made for this example is that the location following the error location is an appropriate return point. This may not always be the case; for example it may be advisable to abort the program in progress and return to a supervisory program.

DMA INTERRUPTS

The direct memory access (DMA) option provides high speed block transfers of data between I/O devices and memory. For the most part DMA operates independently of the interrupt system. (Refer to the description of DMA operation in the Input/Output Section of this manual.)

The only time that DMA generates an interrupt is when it has completed transferring a specified block of data. Since there are two DMA channels, two interrupt locations are reserved for this option: location 00006 (interrupt from DMA channel 1) and location 00007 (interrupt from DMA channel 2). The channel 1 interrupt has priority over the channel 2 interrupt. Since these interrupts are primarily completion signals to the programmer, and are therefore application dependent, no subroutine example is given.

I/O INTERRUPTS

The remaining interrupt locations (octal 00010 through 00077) are available to I/O devices. This represents a total of 56 (decimal) locations, one for each of 56 I/O channels.

In typical input/output operation, the computer issues a programmed command (e.g., set control/clear flag instruction STC,C) to one or more external devices, causing these

Table 5. Sample Memory Protect/Parity Error Subroutine

LABEL	OPCODE	OPERAND	COMMENTS
MPPE	NOP		Memory Protect/Parity Error Subroutine
	CLF	0	Turn off interrupt system to inhibit I/O devices
	CLF	5	Turn off P.E. interrupt during subroutine
	STA	SVA	Save A-register contents
	STB	SVB	Save B-register contents
	LIA	5	Get contents of violation register in MP logic
	SSA		Check bit 15 to determine kind of error
	JMP	PERR	If a 1, go to parity error routine
	JMP	MPTR	If a 0, go to memory protect routine
MPTR	—		User's routine in case of memory protect violation
	—		
	—		
	etc.		
	—		
	—		
	—		
	LDA	SVA	Restore A-register
	LDB	SVB	Restore B-register
	STF	0	Enable interrupt system
	STF	5	Turn on parity error interrupt
	STC	5	Turn on memory protect interrupt
	JMP	MPPE,I	Exit the subroutine
PERR	—		User's routine in case of parity error
	—		
	—		
	etc.		
	—		
	—		
	—		
	JMP	PERR-6	Restore accumulators, turn on interrupts, exit

devices to begin their read or write operation. Each device will put data into (input) or take data from (output) the input/output buffer on each individual interface card. During this time, the computer may continue running a program or may be programmed into a waiting loop to wait for a specific device. On completion of the read or write operation, each device returns an operation completed signal (flag) to the computer. The flags are passed through a priority network which allows only one device to be serviced regardless of the number of flags simultaneously present. The flag with the highest priority generates an interrupt signal at the end of the current machine cycle, except under any of the following circumstances.

a. Interrupt system disabled or device interrupt disabled.

b. JMP indirect or JSB indirect not sufficiently executed. These instructions inhibit all interrupts, except memory protect, until the instruction (plus one phase of the succeeding instruction) is completed, or until at least three indirect references have occurred. The memory protect interrupt for a jump violation will occur on completion of the execute phase, but the jump itself will be inhibited.

c. Instruction in an interrupt location not sufficiently executed, even if of lower priority. Any interrupt inhibits the entire interrupt system until at least two phases have been completed. (JMP indirect and JSB indirect will be fully executed.)

d. Direct memory access option in process of transferring data.

e. The current instruction is one which may affect the priorities of input/output devices (STC, CLC, STF, CLF). The interrupt in this case must wait until the end of the succeeding machine cycle.

A set flag flip-flop inhibits all interrupt requests below it on the priority string (provided that the control flip-flop is also set). Once the flag flip-flop is cleared the next lower device can then interrupt. A service subroutine for any device can be interrupted only by a higher priority device; then, after the higher device is serviced, the interrupted subroutine may continue. In this way, it is possible for several service subroutines to be in a state of interruption at one time; each will be permitted to continue when the higher priority device is serviced. All service subroutines normally end with a JMP indirect instruction to return the computer to the point of interrupt.

For the programmer, communication with I/O devices is simplified by the availability of standard driver routines. Hewlett-Packard furnishes an I/O driver as an accessory to each standard peripheral device supplied by HP. The drivers supplied by HP conform to the design specifications of the HP Basic Control System and are subsequently referred to as BCS drivers. BCS drivers can be integrated into an existing basic control system simply by adding the additional driver to the system in a simple configuration process. BCS drivers generally have the following characteristics:

a. I/O is overlapped with processing using the computer priority interrupt system.

b. Each driver may operate identical devices occupying different I/O locations.

c. Provides status and error information to user and system I/O requests.

d. Compatible with other modules of HP software such as the Input/Output Control (IOC) program and the FORTRAN I/O program called the Formatter.

e. The object code for a BCS driver is relocatable binary.

The modularity of the basic control system provides the user with a very flexible operating system. The functions of the modules can be illustrated by following the sequence of events through a series of I/O transfers. An input transfer is used as an example. See figure 5.

The user or system I/O request is made to a unique entry point in the IOC program. After checking the request for validity, IOC obtains the memory address of the BCS driver for the requested device. Control is transferred to the BCS driver and the input operation is initiated. After initiation the BCS driver transfers control back to the user or system program. The program continues processing until the I/O device completes a single operation. At that time an interrupt request is generated which forces transfer of control to the BCS driver once again. The data is transferred between the device and a specified memory buffer and the I/O device is commanded to do another operation. This process continues until all data has been transferred and the user or system input request is satisfied.

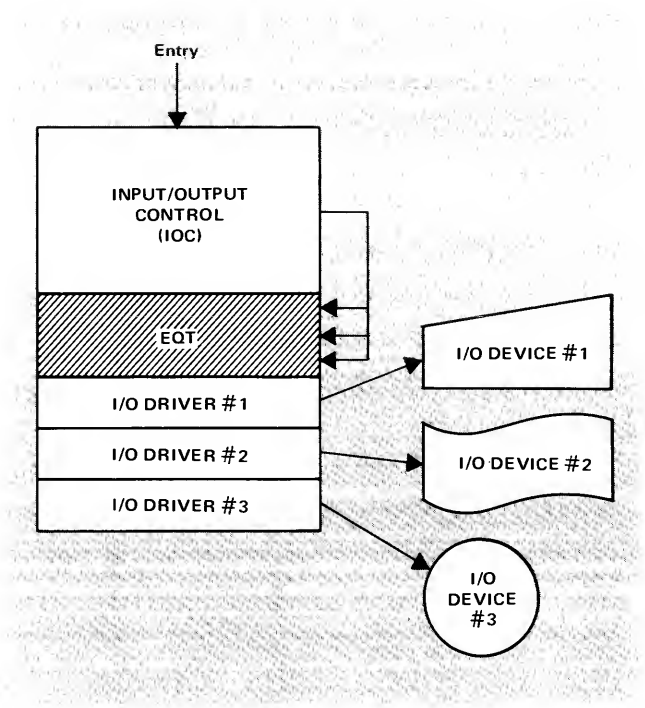


Figure 5. Modules of BCS

The equipment table (EQT) is a memory table created at configuration time to describe the hardware I/O channel of the device, the name and address of the I/O driver to be used, a status word, and a transmission log to be used by the I/O driver. Each physical I/O device (or, sometimes, I/O subsystem consisting of two or more devices) in the system is defined by an entry in the EQT. The EQT provides the interface between IOC and the BCS driver and in addition provides for device independent programming.

CENTRAL INTERRUPT REGISTER

Each time an interrupt occurs, the address of the interrupt location is stored in the central interrupt register. The contents of this register are accessible at any time with a LIA 04 or LIB 04 instruction. This puts the address of the most recent interrupt into the A- or B-register.

INTERRUPT SYSTEM CONTROL

I/O address 00 is a master control address for the interrupt system. A STF 00 instruction enables the entire interrupt system, and a CLF 00 instruction disables the interrupt system. The two exceptions are the power fail interrupt, which cannot be disabled, and parity error interrupt, which can only be selectively enabled or disabled by STF 05 or CLF 05, respectively.

Whenever power is turned on, a clear signal to I/O address 00 automatically disables the interrupt system. The INTERRUPT SYSTEM pushbutton on the operator panel may be used to switch the interrupt system on or off manually. However, programs dependent on interrupt operation should include a STF 00 instruction to ensure that the interrupt system is enabled in the run mode.

This section defines each of the 80 machine instructions of the 2100A Computer. Definitions are grouped according to instruction type: memory reference, register reference, input/output, extended arithmetic memory reference, and extended arithmetic register reference.

With each definition is a diagram showing the machine coding of the instruction. The gray shaded bits code the instruction type and the blue shaded bits code the specific instruction. Unshaded bits are further described under the introduction to each instruction type. The mnemonic code and instruction name are given above each diagram.

In all cases where an additional bit is used to specify a secondary function (D/I, Z/C, or H/C), the choice is made by coding a logic 0 or 1 respectively. That is, a logic 0 codes D, Z, and H, and a logic 1 codes I, C, and C. These abbreviations are defined as follows:

D = Direct addressing
I = Indirect addressing
Z = Zero page
C = Current page
H = Hold flag
C = Clear flag

INSTRUCTION TIMING

All instructions except ISZ and the extended arithmetic instructions are fully executed in 1.96 microseconds. ISZ is executed in 2.94 microseconds, and the extended arithmetic instructions are executed in the times shown in table 6. The Divide instruction executes faster than shown if the divisor is positive (15.68 microseconds) or if overflow occurs (11.76 microseconds). If indirect addressing is used with any of the single-word memory reference instructions, 0.98 microsecond is added for each level of indirect addressing used; 1.96 microseconds are added for each level of indirect addressing with extended arithmetic memory reference instructions.

Instructions are executed in two or more phases. The first phase is the fetch phase, which obtains an instruction from memory and transfers it into the central processor's instruction register. Next, there can be one or more indirect phases. The indirect phase, which applies only to single-length memory reference instructions, obtains a new operand address for the same (current) instruction. Lastly there is an execute phase, which accomplishes actual execution of the instruction. For extended arithmetic memory reference instructions, indirect addressing is also accomplished in the execute phase. Although the duration of a phase varies considerably (from 588 nanoseconds to an indeterminate time in the case of extended arithmetic indirect addressing), synchronization with memory or input/output operations results in overall execution times as specified in the preceding paragraph.

Table 6. Extended Arithmetic Execution Times

INSTRUCTION		TIME (μsec)
MPY (Multiply)		10.78
DIV (Divide)	Max	16.66
DLD (Double Load)		5.88
DST (Double Store)		5.88
	Number of Shifts	
ASR (Arithmetic Shift Right)	1, 2, 3	2.94
	4, 5, 6, 7, 8	3.92
	9, 10, 11, 12, 13	4.90
	14, 15, 16	5.88
ASL (Arithmetic Shift Left)	1, 2, 3, 4, 5	4.90
	6, 7, 8, 9, 10	5.88
	11, 12, 13, 14, 15	6.86
	16	7.84
LSR, RRR (Logical Shift Right, Rotate Right)	1, 2	2.94
	3, 4, 5, 6, 7	3.92
	8, 9, 10, 11, 12	4.90
	13, 14, 15, 16	5.88
LSL, RRL (Logical Shift Left, Rotate Left)	1, 2, 3, 4	4.90
	5, 6, 7, 8, 9	5.88
	10, 11, 12, 13, 14	6.86
	15, 16	7.84

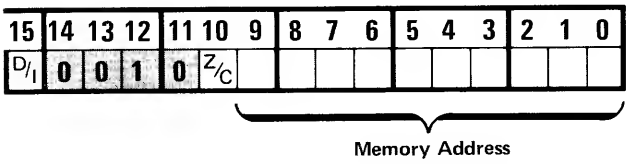
MEMORY REFERENCE INSTRUCTIONS

The 14 memory reference instructions execute a function involving data in memory. Bits 0 through 9 specify the affected memory location on a given memory page, or, if indirect addressing is used, the next address to be referenced. Indirect addressing may be continued to any number

of levels; when the D/I bit is "0" (specifying direct addressing), that location will be taken as the effective address. The A- and B-registers may be addressed as locations 00000 and 00001 (octal) respectively.

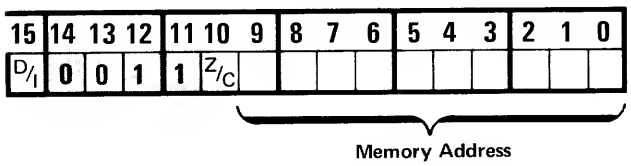
If bit 10 (Z/C) is a "0", the memory address is on page zero; if bit 10 is a "1", the memory address is on the current page. If the A- or B-register is addressed, bit 10 must be a "0" to specify page zero, unless the current page is page zero.

AND "AND" TO A



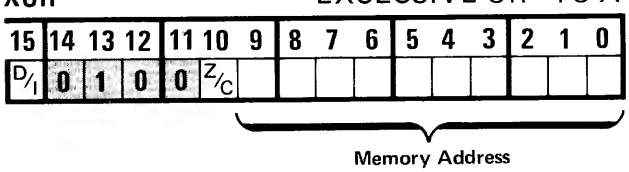
The contents of the addressed location are logically "anded" to the contents of the A-register. The contents of the memory cell are left unaltered.

JSB JUMP TO SUBROUTINE



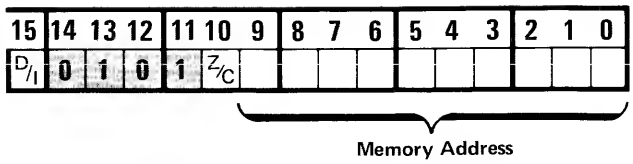
This instruction, executed in location P, causes computer control to jump unconditionally to the memory location (m) specified in the address portion of the JSB instruction word. The contents of the P-register plus one (return address) is stored in location m, and the next instruction to be executed will be that contained in the next location (m + 1). A return to the main program sequence at P + 1 may be effected by a jump indirect through location m.

XOR "EXCLUSIVE OR" TO A



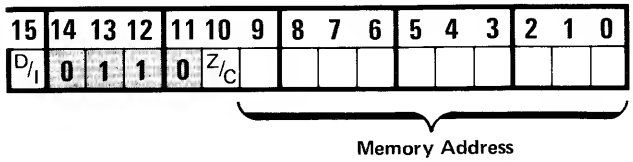
The contents of the addressed location are combined with the contents of the A-register as an "exclusive or" logic operation. The contents of the memory cell are left unaltered.

JMP JUMP



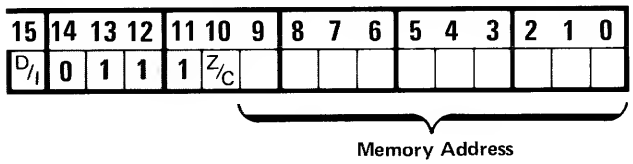
The instruction transfers control to the addressed location. That is, JMP causes the P-register to be set according to the memory address portion of the instruction word, so that the next instruction will be read from that location.

IOR "INCLUSIVE OR" TO A



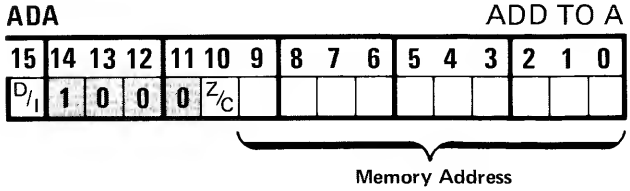
The contents of the addressed location are combined with the contents of the A-register as an "inclusive or" logic operation. The contents of the memory cell are left unaltered.

ISZ INCREMENT AND SKIP IF ZERO



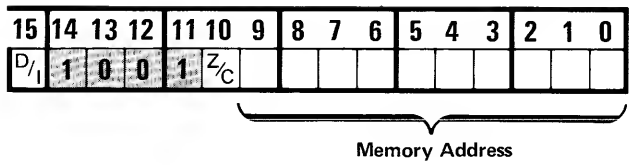
An ISZ instruction adds one to the contents of the addressed memory location. If the result of this operation is zero, the next instruction is skipped; i.e., the P-register is advanced by two instead of one. Otherwise, the program proceeds normally to the next instruction in sequence. The incremented value is written back into the memory cell in either case. An ISZ instruction referencing locations zero or one (A- or B-register) cannot cause setting of the extend or overflow bits (unlike INA and INB).

ADA ADD TO A



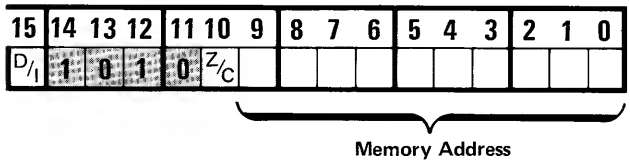
The contents of the addressed memory location are added to the contents of the A-register, and the sum remains in the A-register. The result of the addtion may set the extend or overflow bits. The contents of the memory cell are unaltered.

ADB ADD TO B



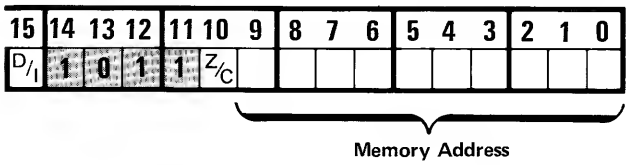
The contents of the addressed memory location are added to the contents of the B-register, and the sum remains in the B-register. Extend or overflow bits may be set, as for ADA. The contents of the memory cell are unaltered.

CPA COMPARE TO A



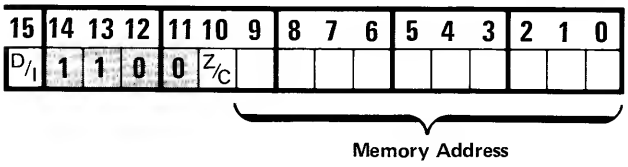
The contents of the addressed location are compared with the contents of the A-register. If the two 16-bit words are unequal, the next instruction is skipped; i.e., the P-register is advanced by two instead of one. If the words are identical, the program proceeds normally to the next instruction in sequence. The contents of neither the A-register nor the memory cell are altered.

CPB COMPARE TO B



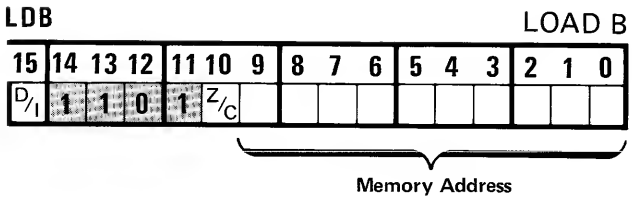
Same as CPA, except comparison is made with the B-register.

LDA LOAD A



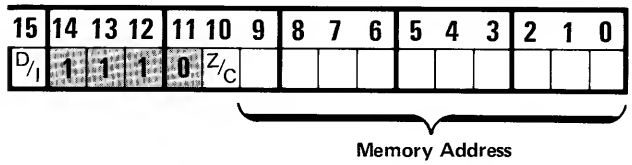
The A-register is loaded with the contents of the addressed location. The contents of the memory cell are unaltered.

LDB LOAD B



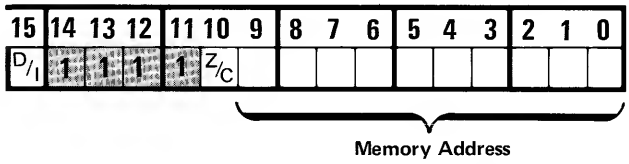
The B-register is loaded with the contents of the addressed location. The contents of the memory cell are unaltered.

STA STORE A



The contents of the A-register are stored in the addressed location. The previous contents of the memory cell are lost; the A-register is unaltered.

STB STORE B



The contents of the B-register are stored in the addressed location. The previous contents of the memory cell are lost; the B-register is unaltered.

REGISTER REFERENCE INSTRUCTIONS

The 39 register reference instructions execute various functions on data contained in the A- B- and E-registers. The instructions are divided into two groups: the shift-rotate group and the alter-skip group. In each group, several instructions may be combined into one word and are thus individually termed microinstructions. Since the two groups are separate and distinct, microinstructions from the two groups cannot be mixed. Unshaded bits in the coding diagrams are available for combining other microinstructions.

SHIFT-ROTATE GROUP. The 20 instructions of the shift-rotate group are defined first. A comparison of shift and rotate functions is given in figure 6. Rules for combining microinstructions are as follows. (Refer to table 7.)

a. Only one microinstruction can be chosen from the multiple-choice columns.

b. References to both A- and B-registers cannot be mixed.

c. The sequence of execution is left to right.

d. In machine code, use zeros to exclude unwanted micro-instruction bits.

e. Use a "1" bit in bit 9 to enable shifts or rotates in the first position, and a "1" bit in bit 4 to enable shifts or rotates in the second position.

f. The extend bit is not affected unless specifically stated. However, if a rotate-with-E instruction (ERA/B, ELA/B) is coded but disabled by a "0" in bit 9 or 4, the E-register will be updated even though the A- or B-register is not affected; code a NOP (three zeros) to avoid this situation.

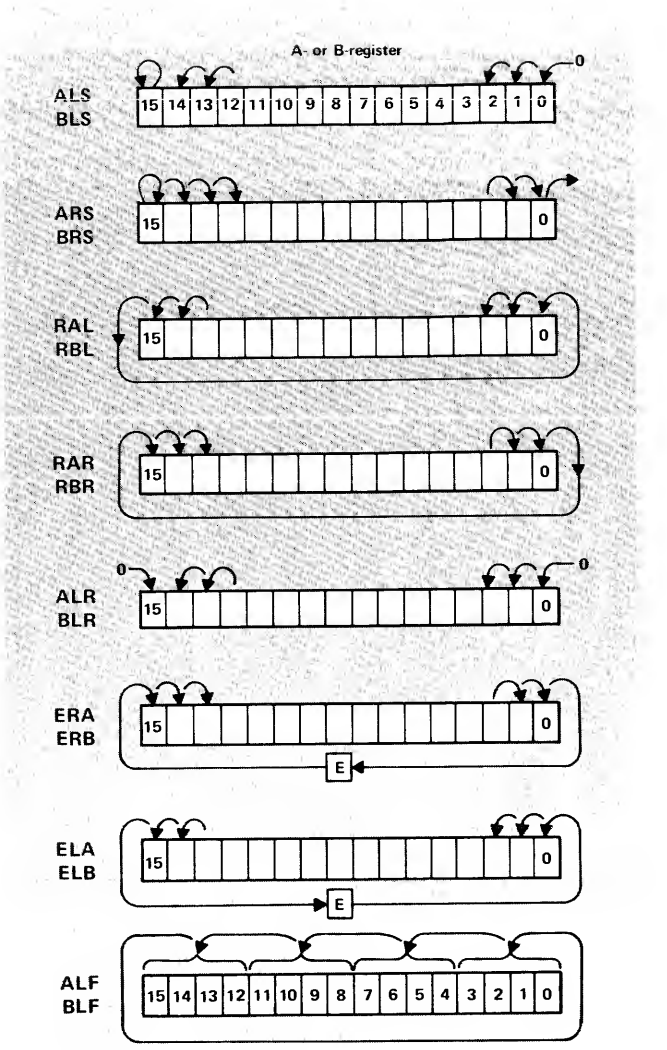


Figure 6. Shift and Rotate Functions

Table 7. Shift-Rotate Combining Guide

ALS, ARS, RAL, RAR, ALR, ALF, ERA, ELA	[,CLE]	[,SLA]	ALS, ARS, RAL, RAR, ALR, ALF, ERA, ELA
BLS, BRS, RBL, RBR, BLR, BLF, ERB, ELB	[,CLE]	[,SLB]	BLS, BRS, RBL, RBR, BLR, BLF, ERB, ELB

CLE — CLEAR E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0		0					1					

Clear E-register (extend bit).

NOP — NO OPERATION

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

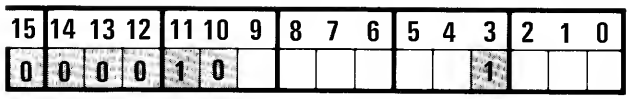
An all-zero instruction word causes a no-operation cycle.

SLA — SKIP IF LSB OF A IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0							1			

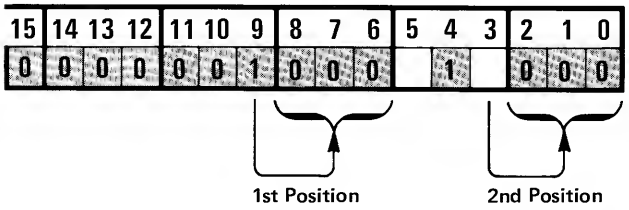
The next instruction is skipped if the least significant bit of the A-register is "0".

SLB SKIP IF LSB OF B IS ZERO



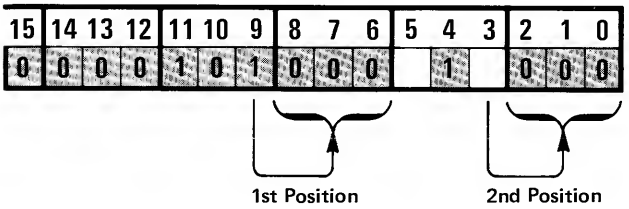
The next instruction is skipped if the least significant bit of the B-register is "0".

ALS A LEFT SHIFT



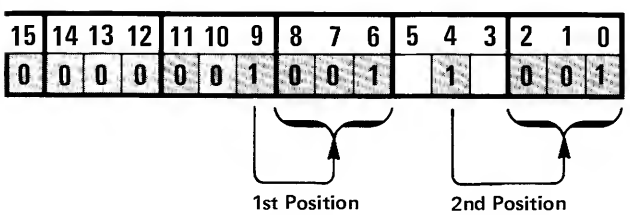
The A-register is arithmetically shifted left one place, 15 magnitude bits only. Bit 15 (sign bit) is not affected; bit shifted out of bit 14 is lost. A "0" replaces vacated bit 0.

BLS B LEFT SHIFT



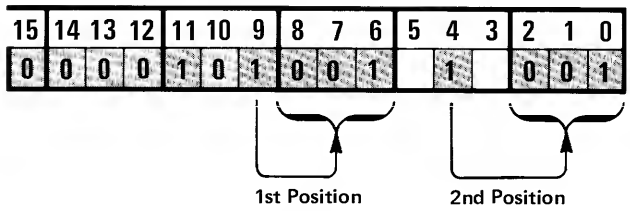
The B-register is arithmetically shifted left one place, 15 magnitude bits only. Bit 15 (sign bit) is not affected; bit shifted out of bit 14 is lost. A "0" replaces vacated bit 0.

ARS A RIGHT SHIFT



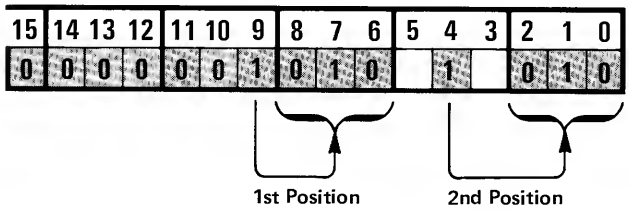
The A-register is arithmetically shifted right one place, 15 magnitude bits only. Bit 15 (sign bit) is not affected; copy of sign bit is shifted into bit 14. Bit shifted out of bit 0 is lost.

BRS B RIGHT SHIFT



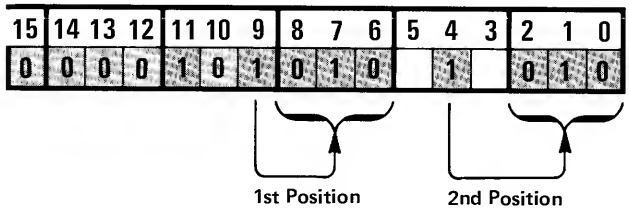
The B-register is arithmetically shifted right one place, 15 magnitude bits only. Bit 15 (sign bit) is not affected; copy of sign bit is shifted into bit 14. Bit shifted out of bit 0 is lost.

RAL ROTATE A LEFT



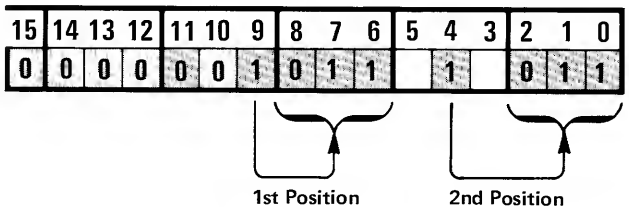
Rotate A-register left one place, all 16 bits. Bit 15 is rotated around to bit 0.

RBL ROTATE B LEFT



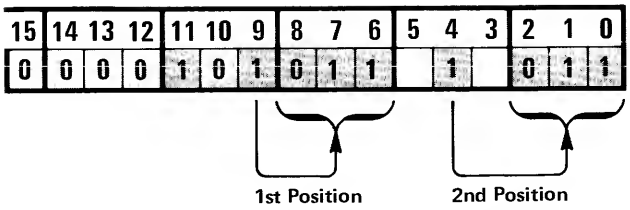
Rotate B-register left one place, all 16 bits. Bit 15 is rotated around to bit 0.

RAR ROTATE A RIGHT



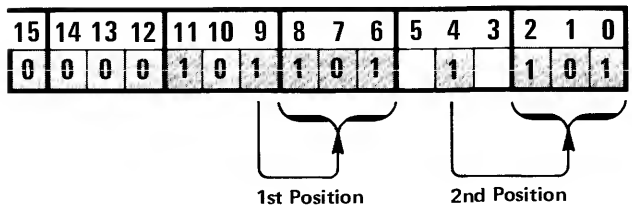
Rotate A-register right one place, all 16 bits. Bit 0 is rotated around to bit 15.

RBR ROTATE B RIGHT



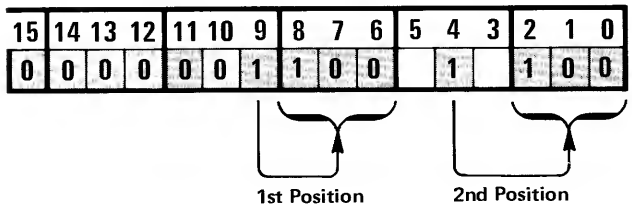
Rotate B-register right one place, all 16 bits. Bit 0 is rotated around to bit 15.

ERB ROTATE E RIGHT WITH B



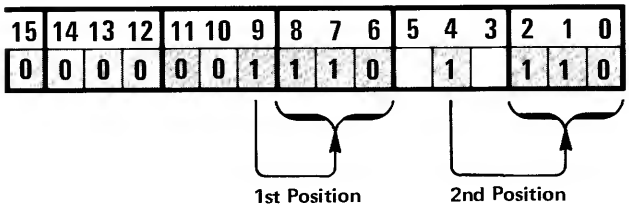
Rotate E-register right with B-register, one place (17 bits). Bit 0 is rotated into extend register; extend content is rotated into bit 15.

ALR A LEFT SHIFT, CLEAR SIGN



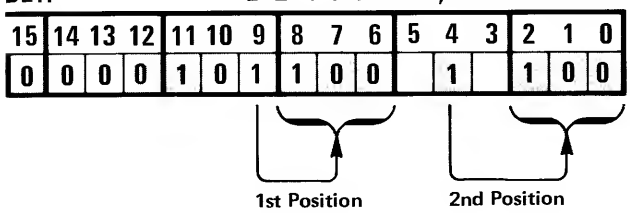
Shift A-register left one place, same as ALS, but clear sign bit after shift.

ELA ROTATE E LEFT WITH A



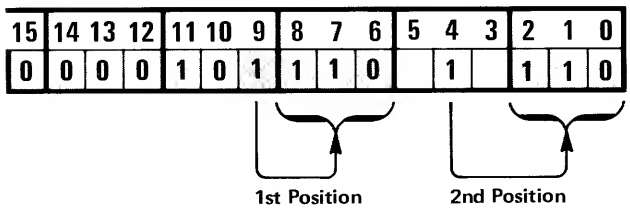
Rotate E-register left with A-register, one place (17 bits). Bit 15 is rotated into extend register; extend content is rotated into bit 0.

BLR B LEFT SHIFT, CLEAR SIGN



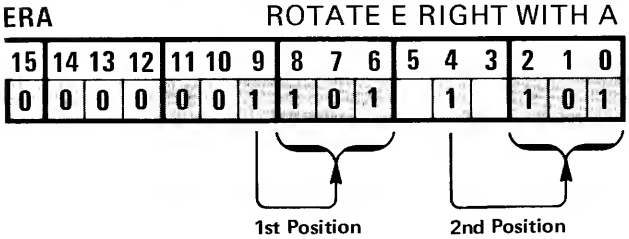
Shift B-register left one place, same as BLS, but clear sign bit after shift.

ELB ROTATE E LEFT WITH B



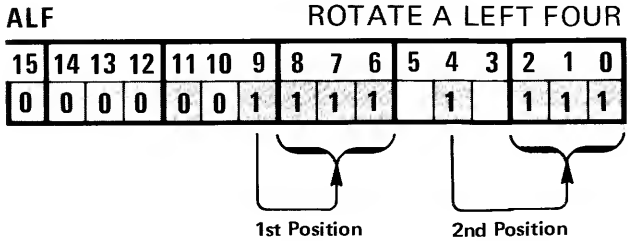
Rotate E-register left with B-register, one place (17 bits). Bit 15 is rotated into extend register; extend content is rotated into bit 0.

ERA ROTATE E RIGHT WITH A



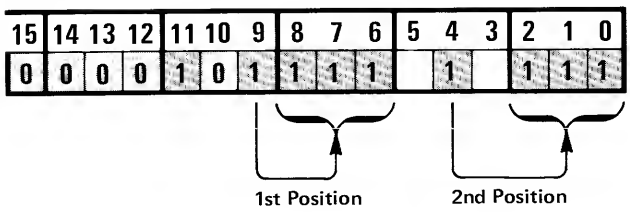
Rotate E-register right with A-register, one place (17 bits). Bit 0 is rotated into extend register; extend content is rotated into bit 15.

ALF ROTATE A LEFT FOUR



Rotate A-register left four places, all 16 bits. Bits 15, 14, 13, 12 are rotated around to bits 3, 2, 1, 0 respectively. Equivalent to four successive RAL instructions.

BLF ROTATE B LEFT FOUR



Rotate B-register left four places, all 16 bits. Bits 15, 14, 13, 12 are rotated around to bits 3, 2, 1, 0 respectively. Equivalent to four successive RBL instructions.

ALTER-SKIP GROUP. The 19 instructions of the alter-skip groups are defined next. This group is specified by a "1" bit in bit 10. Rules for combining microinstructions are as follows. (Refer to table 8.)

a. Only one microinstruction can be chosen from the multi-choice columns.

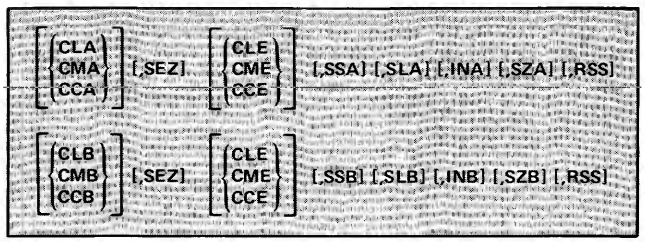
b. References to both A- and B-registers cannot be mixed.

c. The sequence of execution is left to right.

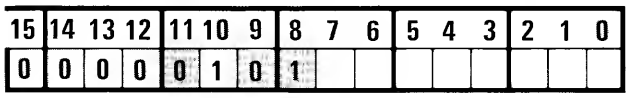
d. If two or more skip functions are combined, the skip will occur if either or both conditions are met. One exception exists: refer to RSS instruction.

e. In machine code, use zeros to exclude unwanted micro-instruction bits.

Table 8. Alter-Skip Combining Guide

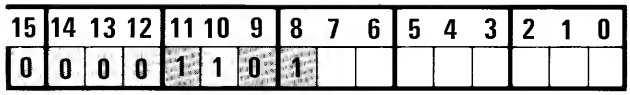


CLA CLEAR A



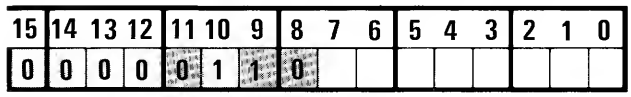
Clear the A-register.

CLB CLEAR B



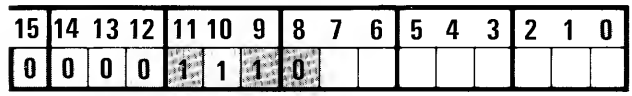
Clear the B-register.

CMA COMPLEMENT A



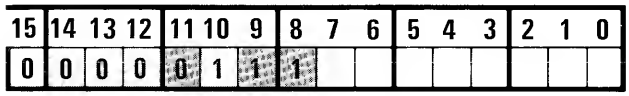
Complement the A-register. (One's complement.)

CMB COMPLEMENT B



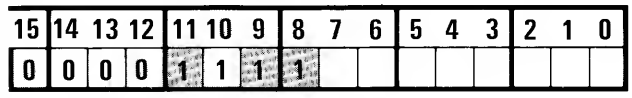
Complement the B-register. (One's complement.)

CCA CLEAR AND COMPLEMENT A



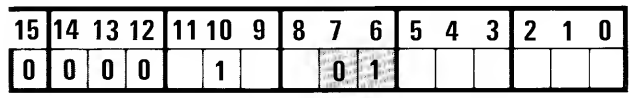
Clear, then complement the A-register. Puts 16 ones in the A-register; this is the two's complement form of –1.

CCB CLEAR AND COMPLEMENT B



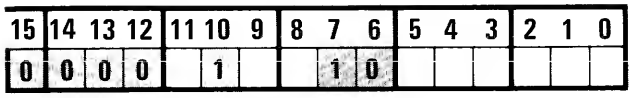
Clear, then complement the B-register. Puts 16 ones in the B-register; this is the two's complement form of –1.

CLE CLEAR E



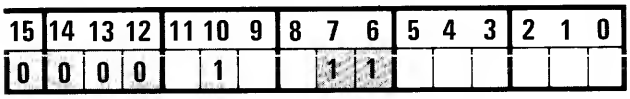
Clear the E-register (extend bit).

CME COMPLEMENT E



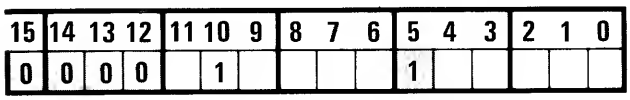
Complement the E-register (extend bit).

CCE CLEAR AND COMPLEMENT E



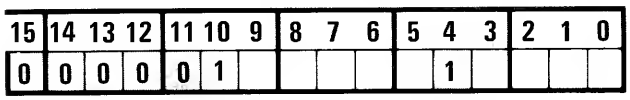
Clear, then complement the E-register (extend bit). Sets the extend bit to "1".

SEZ SKIP IF E IS ZERO



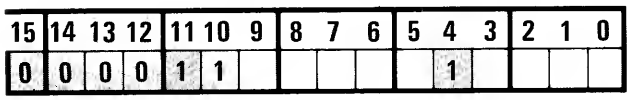
Skip the next instruction if the E-register (extend bit) is zero.

SSA SKIP IF SIGN OF A IS ZERO



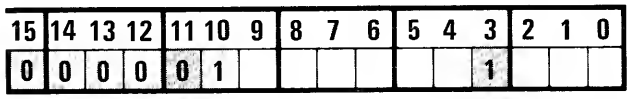
Skip next instruction if the sign bit (bit 15) of the A-register is zero; i.e., skip if the content of A is positive.

SSB SKIP IF SIGN OF B IS ZERO



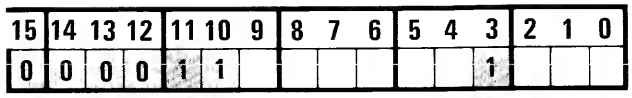
Skip next instruction if the sign bit (bit 15) of the B-register is zero; i.e., skip if the content of B is positive.

SLA SKIP IF LSB OF A IS ZERO



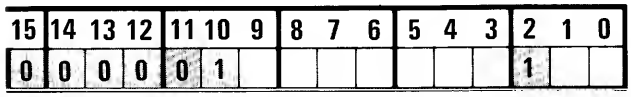
Skip next instruction if the least significant bit of the A-register is zero (i.e., skip if an even number is in A).

SLB SKIP IF LSB OF B IS ZERO



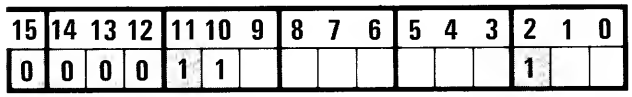
Skip next instruction if the least significant bit of the B-register is zero (i.e., skip if an even number is in B).

INA INCREMENT A



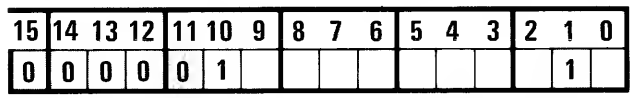
Increment the A-register by one. Can cause setting of extend or overflow bits.

INB INCREMENT B



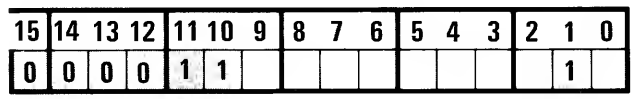
Increment the B-register by one. Can cause setting of extend or overflow bits.

SZA SKIP IF A IS ZERO



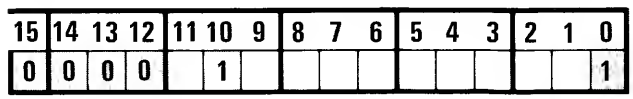
Skip next instruction if the A-register is zero (16 zeros).

SZB SKIP IF B IS ZERO



Skip next instruction if B-register is zero (16 zeros).

RSS REVERSE SKIP SENSE



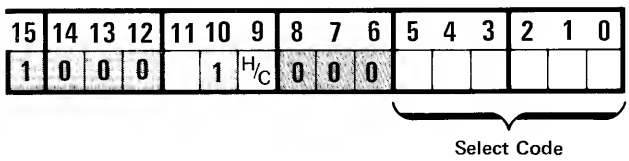
Skip occurs for any of the preceding skip instructions, if present, when the non-zero condition is met. RSS without a skip instruction in the word causes an unconditional skip. If a word with RSS also includes both SSA/B and SLA/B, bits 15 and 0 must both be one for skip to occur; in all other cases the skip occurs if one or more skip conditions are met.

INPUT/OUTPUT INSTRUCTIONS

The 17 input/output instructions provide the capability to set or clear the I/O flag and control bits and the overflow bit, to test the state of the overflow and I/O flag bits, and to transfer data between an I/O channel and the A- and B-registers. In addition, specific instructions in this group control the interrupt system and can cause a programmed halt.

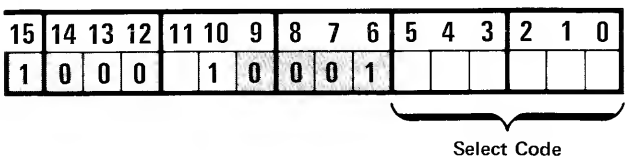
Bit 11, where relevant, specifies the A- or B-register or distinguishes between set control and clear control; otherwise it may be "1" or "0" without affecting the instruction (although the assembler will assign zeros, as shown). Bit 9, where not specified, offers the choice of holding (0) or clearing (1) the device flag after execution of the instruction. (Exception: the H/C bit associated with the last two instructions in this list holds or clears the overflow bit instead of a flag bit.) Bits 8, 7, and 6 identify the instruction. Bits 5 through 0 (unshaded) form select codes to make the instruction apply to one of up to 64 input/output devices or functions.

HLT HALT



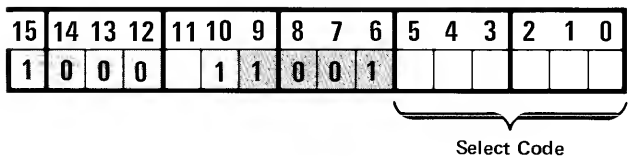
Halts the computer and holds or clears the flag (according to bit 9) of any desired input/output device (bits 5 through 0). The HLT instruction has the same effect as the HALT pushbutton: the HALT switch lights, and the front-panel control switches are enabled. The HLT instruction will be displayed (MEMORY DATA is automatically selected when computer halts), and the P-register will normally indicate the halt location plus one.

STF SET FLAG



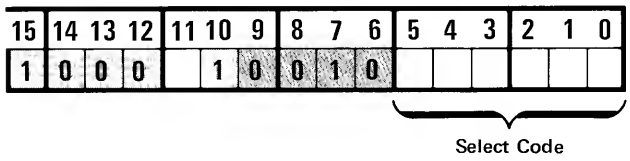
Sets the flag of the selected I/O channel or function. A STF 00 instruction enables the interrupt system for all select codes (except power fail and parity error, which are always enabled).

CLF CLEAR FLAG



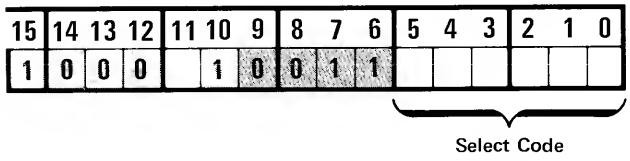
Clears the flag of the selected I/O channel or function. A CLF 00 instruction disables the interrupt system for all select codes (except power fail and parity error, which are always enabled); this does not affect the status of the individual channel flags.

SFC SKIP IF FLAG CLEAR



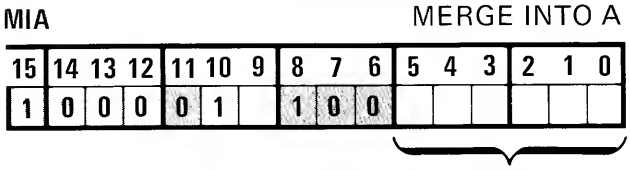
Skip next instruction if the flag of the selected channel is clear (device busy).

SFS SKIP IF FLAG SET



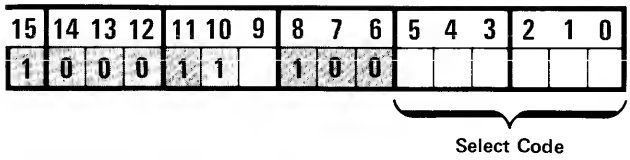
Skip next instruction if the flag of the selected channel is set (device ready).

MIA MERGE INTO A



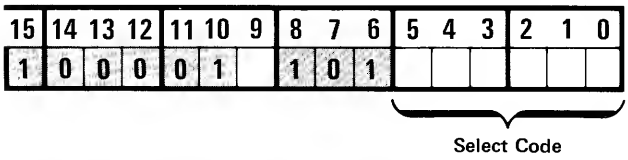
The contents of the input/output buffer associated with the selected device are merged ("inclusive or") into the A-register.

MIB MERGE INTO B



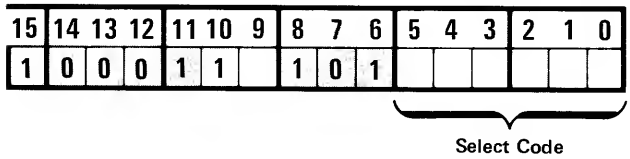
The contents of the input/output buffer associated with the selected device are merged ("inclusive or") into the B-register.

LIA LOAD INTO A



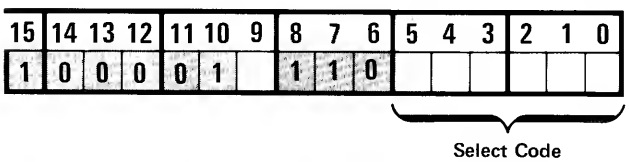
The contents of the input/output buffer associated with the selected device are loaded into the A-register.

LIB LOAD INTO B



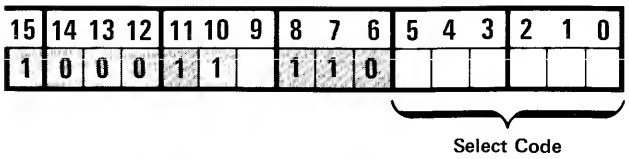
The contents of the input/output buffer associated with the selected device are loaded into the B-register.

OTA OUTPUT A



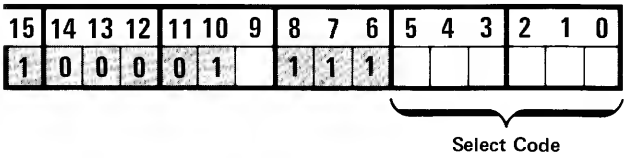
The contents of the A-register are loaded into the input/output buffer associated with the selected device. If the buffer is less than 16 bits in length, the least significant bits of the A-register normally are loaded. (Some exceptions exist, depending on the type of output device.) A-register contents are not altered.

OTB OUTPUT B



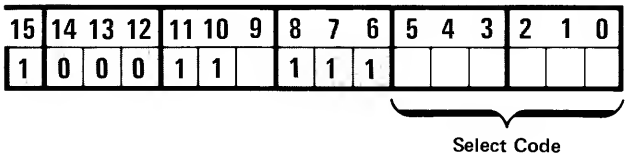
The contents of the B-register are loaded into the input/output buffer associated with the selected device.

STC SET CONTROL



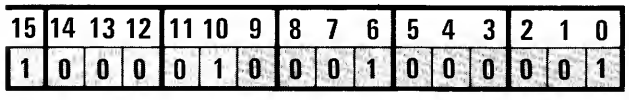
Sets the control bit of the selected I/O channel or function.

CLC CLEAR CONTROL



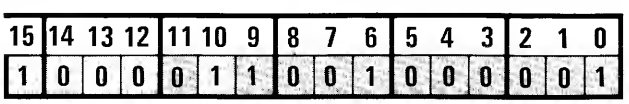
Clears the control bit of the selected I/O channel or function. This turns off a device channel and prevents it from interrupting. A CLC 00 instruction clears all control bits from select code 06 and up, effectively turning off all I/O devices.

STO SET OVERFLOW



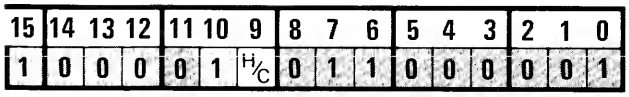
Sets the overflow bit.

CLO CLEAR OVERFLOW



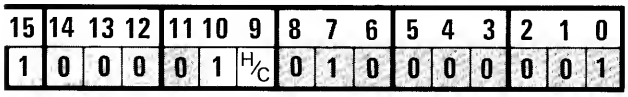
Clears the overflow bit.

SOS SKIP IF OVERFLOW SET



If the overflow register is set, the next instruction of the program is skipped. Use of the H/C bit will hold or clear the overflow bit following execution of this instruction (whether the skip is taken or not).

SOC SKIP IF OVERFLOW CLEAR



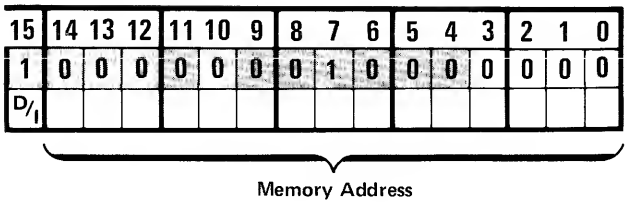
If the overflow register is clear, the next instruction of the program is skipped. Use of the H/C bit will hold or clear the overflow bit following execution of this instruction (whether the skip is taken or not).

EXTENDED ARITHMETIC MEMORY REFERENCE INSTRUCTIONS

The four extended arithmetic memory reference instructions provide for integer multiply and divide, and for loading and storing double-length words to and from the accumulators. The complete instruction requires two words: one for the instruction code, and one for the address. When stored in memory the instruction word is the first to be fetched; the address word is in the next higher location.

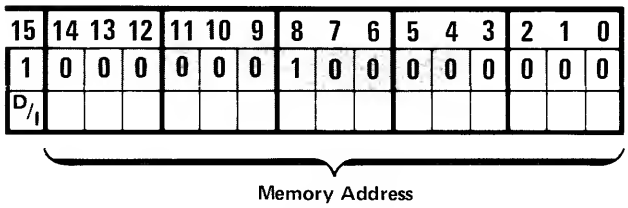
Since 15 bits are available for the address, these instructions may directly address any location in memory. As for all memory reference instructions, indirect addressing to any number of levels may also be used. A "0" in the D/I bit specifies direct addressing; a "1" specifies indirect addressing.

MPY MULTIPLY



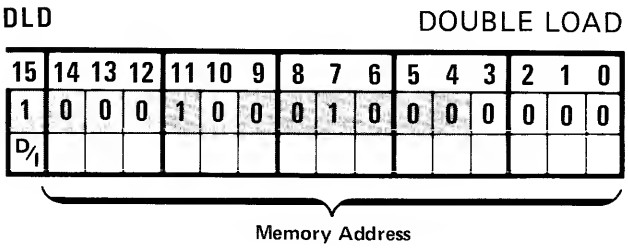
Multiplies a 16-bit integer in the A-register by a 16-bit integer in the addressed memory location. The resulting double-length integer product resides in the B- and A-registers, with the B-register containing the sign bit and most significant 15 bits of the quantity. The A-register may be used as an operand (i.e., memory address 0), resulting in an arithmetic square. Overflow cannot occur; the instruction clears the overflow bit.

DIV DIVIDE



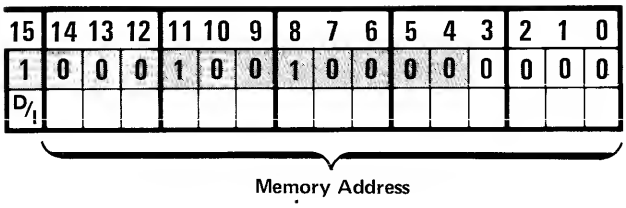
Divides a doubleword integer in the combined B- and A-registers by a 16-bit integer in the addressed memory location. The result is a 16-bit integer quotient in the A-register and a 16-bit integer remainder in the B-register. Overflow can result from an attempt to divide by zero, or from an attempt to divide by a number too small for the dividend. In the former case (divide by zero) execution will be attempted with unpredictable results left in the B- and A-registers. In the latter case (divisor too small) the division will not be attempted and the B- and A-register contents will be unchanged, except that a negative quantity will be made positive. If there is no divide error, the overflow bit is cleared.

DLD DOUBLE LOAD



Loads the contents of addressed memory location m (and m+1) into the A- and B-registers, respectively.

DST DOUBLE STORE



Stores the doubleword quantity in the A- and B-registers into addressed memory location m (and m+1), respectively.

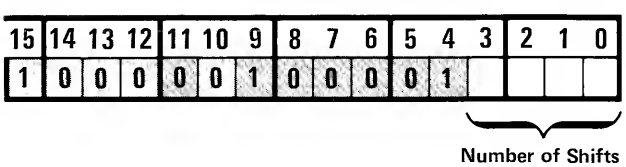
EXTENDED ARITHMETIC REGISTER REFERENCE INSTRUCTIONS

The six extended arithmetic register reference instructions provide various types of shifting operations on the combined contents of the B- and A-registers. The B-register is considered to be on the left (most significant word) and the A-register is considered to be on the right (least significant word). An example of each type of shift operation is illustrated in figure 7.

The complete instruction is given in one word and includes four bits (unshaded) to specify the number of shifts, from 1 to 16. By viewing the four bits as a binary-coded number, the number of shifts is easily expressed; e.g., binary-coded 1 for one shift, binary-coded 2 for two shifts, etc. The maximum of 16 shifts is coded with four zeros; this essentially exchanges the B- and A-register contents.

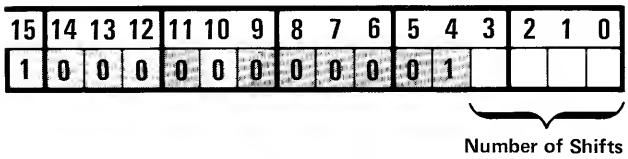
The extend bit is not affected by any of the following instructions. Except for the arithmetic shifts, overflow also is not affected.

ASR ARITHMETIC SHIFT RIGHT



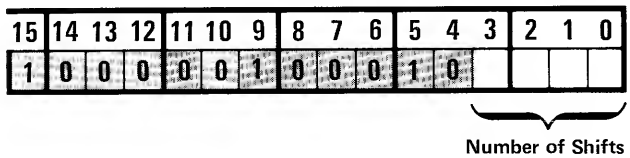
Arithmetically shifts the combined contents of the B- and A-registers right, n places. The value of n may be any number from 1 through 16. The sign bit is unchanged and is extended into bit positions vacated by the right shift. Data bits shifted out of the least significant end of the A-register are lost. Overflow cannot occur; the instruction clears the overflow bit.

ASL ARITHMETIC SHIFT LEFT



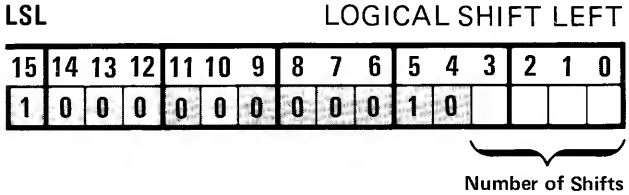
Arithmetically shifts the combined contents of the B- and A-registers left, n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated low order positions of the A-register. The sign bit is unchanged, and data bits are lost out of bit 14 of the B-register. If one of the bits lost is a significant data bit ("1" for positive numbers, "0" for negative numbers), overflow will be set; otherwise, overflow will be cleared during execution. See ASL example in figure 7. Note that two additional shifts in this example would cause an error by losing a significant "1".

LSR LOGICAL SHIFT RIGHT



Logically shifts the combined contents of the B- and A-registers right, n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated high order bit positions of the B-register, and data bits are lost out of the low order bit positions of the A-register.

LSL LOGICAL SHIFT LEFT



Logically shifts the combined contents of the B- and A-registers left, n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated low order bit positions of the A-register, and data bits are lost out of the high order bit positions of the B-register.

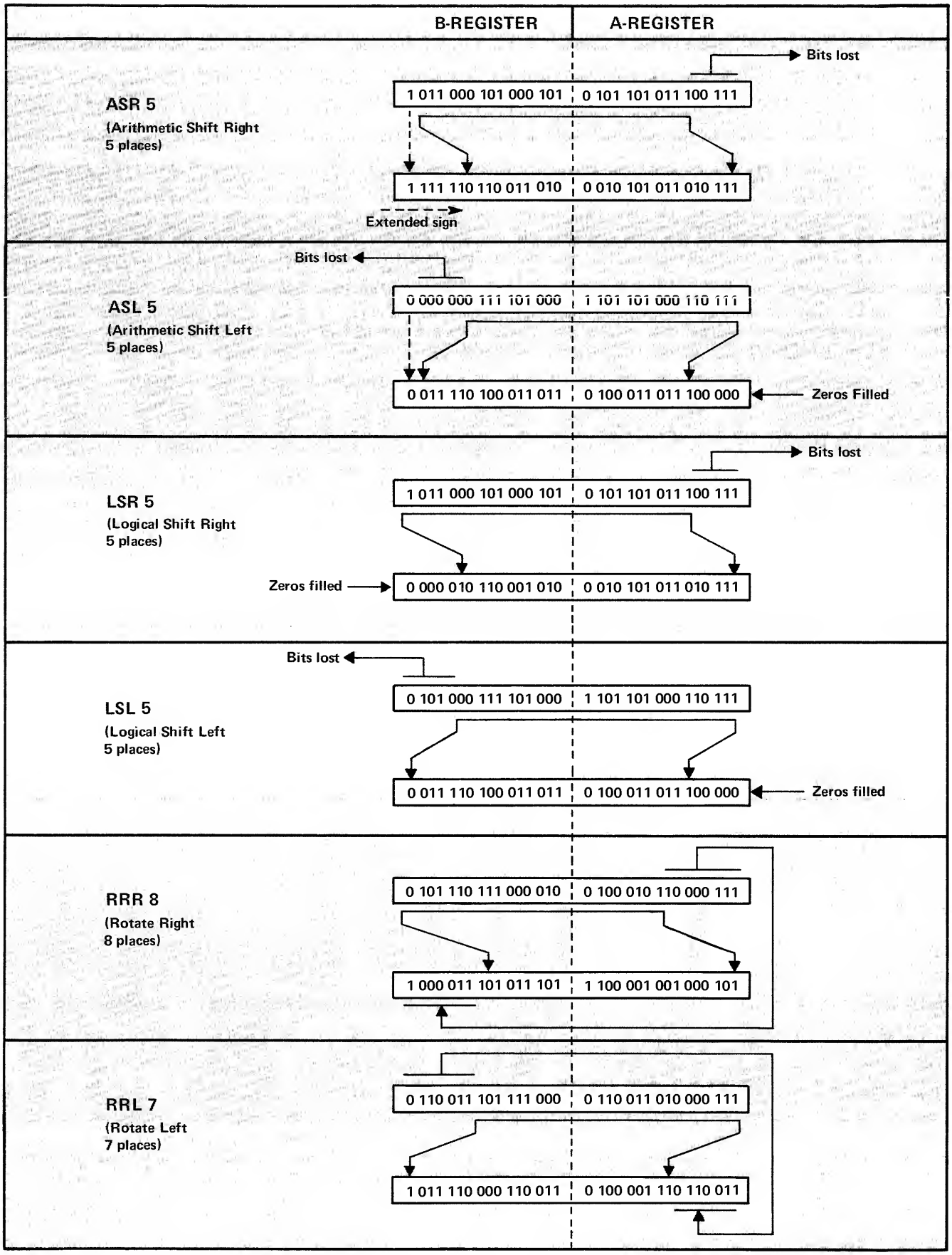
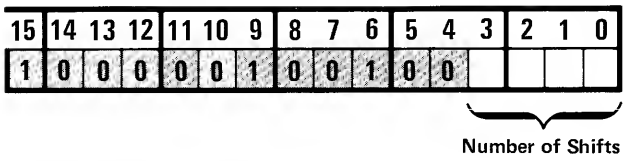


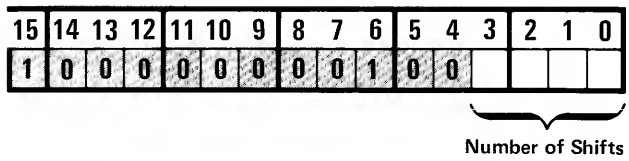
Figure 7. Examples of Doubleword Shifts and Rotates

RRR ROTATE RIGHT



Rotates the combined contents of the B- and A-registers right, n places. The value of n may be any number from 1 through 16. No bits are lost or filled in. Data bits shifted out of the low order end of the A-register are rotated around to enter the high order end of the B-register.

RRL ROTATE LEFT



Rotates the combined contents of the B- and A-registers left, n places. The value of n may be any number from 1 through 16. No bits are lost or filled in. Data bits shifted out of the high order end of the B-register are rotated around to enter the low order end of the A-register.

The purpose of the input/output system is to transfer data between the computer and external devices.

Normally (see figure 8), data is transferred through the A- or B-register. This type of transfer occurs in three distinct steps:

a. between external device and its interface card in the computer;

b. between the interface card and the A- or B-register; and

c. between the A- or B-register and memory.

This three-step process applies to both the "in" direction (as above) and the "out" direction (reverse order). This type of transfer, which is executed under program control, allows the computer logic to manipulate the data during the transfer process.

Optionally (also shown in figure 8), data may be transferred automatically under control of the direct memory access (DMA) option. Once the DMA option has been initialized, no programming is involved, and the transfer is reduced to a two-step process: the transfer between the device and its interface, and the transfer between the interface and memory. Two DMA channels are provided and are assignable to operate with any two device interfaces.

Since the DMA transfer eliminates programmed loading and storing via the accumulators, the time involved is very short. Thus DMA is used with high-speed devices capable of transferring data at rates up to 1,020,400 sixteen-bit words per second. Further information on the direct memory access option is given later in this section.

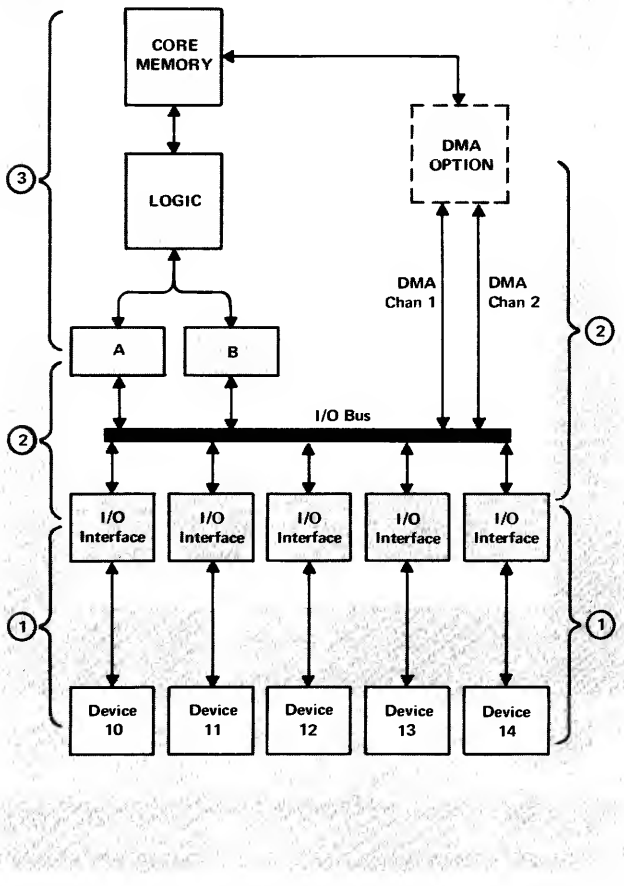


Figure 8. Input/Output System

I/O ADDRESSING

As shown in figure 9, an external device is connected by a cable directly to an interface card located inside the computer. The interface card, in turn, plugs into one of the 14 input/output slots. Each slot is assigned a fixed address, called the select code. The computer can then communicate with a specific device on the basis of its select code.

Figure 9 shows an interface card being inserted into the I/O slot having the highest priority. This slot is assigned select code 10 (octal). If it is decided that the associated device should have lower priority, its interface and cable may be exchanged with those occupying some other I/O slot. This will change both the priority and the I/O address. However, due to priority chaining (explained later), there can be no vacant slots from select code 10 to the highest used select code (if the interrupt mode is to be used).

Only select codes 10 through 77 (octal) are available for input/output devices. The lower select codes (00 through 07) are reserved for other features discussed elsewhere in this manual. As figure 9 shows, select codes 10 through 25 are available in the mainframe of the computer. If an I/O extender is used, slot 25 is used for interconnection of the extender, and select codes 25 through 65 will then be available in the extender. This is a total of 45 (decimal) select codes. The full range of 56 select codes (10 through 77, octal) are available using the multiplexer option. The multiplexer card may be plugged into any slot, but the rule that there can be no vacant slots (select codes) from 10 upward must be maintained.

In some cases, certain devices may require two I/O slots and two select codes. This requirement is fully explained in documentation supplied with the applicable interface.

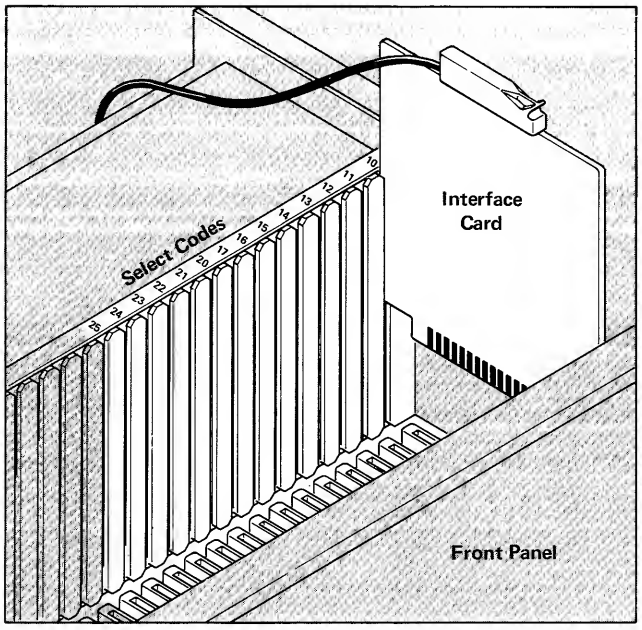


Figure 9. I/O Address Assignments

I/O PRIORITY

When a device is ready to be serviced (refer to "I/O Data Transfer"), it causes its interface to request an interrupt so that the computer will interrupt the current program and service the device. Since many device interfaces will be requesting service at random times, it is necessary to establish an orderly sequence for granting interrupts. Secondly, it is desirable that high speed devices should not have to wait for low speed device transfers.

Both of these requirements are met by a series-linked priority structure, illustrated in simplified form in figure 10. The bold line, representing a priority enabling signal, is routed in series through each card which is capable of causing an interrupt. The card may not interrupt unless this enabling signal is present at its input.

Each device (or other interrupt function) can break the enabling line when it requests an interrupt. If two devices simultaneously request an interrupt, obviously the device with the lowest select code number will be the first one which can interrupt, since it has broken the enable line for the higher select codes. The other device cannot begin its service routine until the first device is finished; however a still higher priority device (lower select code) may interrupt the service routine of the first device.

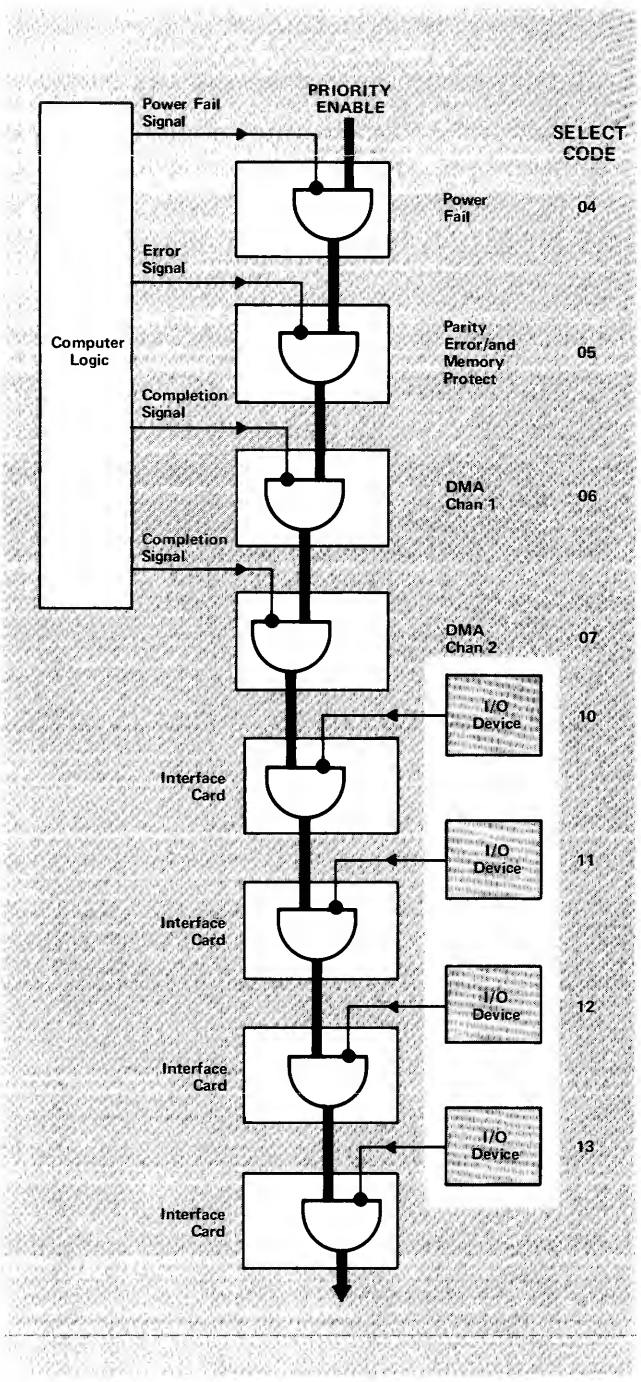


Figure 10. Priority Linkage

Figure 11 illustrates a hypothetical case in which several devices require servicing by interrupting a CPU program. Both simultaneous and time-separated interrupt requests are considered.

Assume that the computer is running a CPU program when an interrupt from I/O channel 12 occurs (at reference time t1). A JSB instruction in the interrupt location for select code 12 causes a program jump to the service routine for the channel 12 device. The JSB instruction automatically saves the return address (in a location which the programmer must reserve in his routine) for a later return to the CPU program.

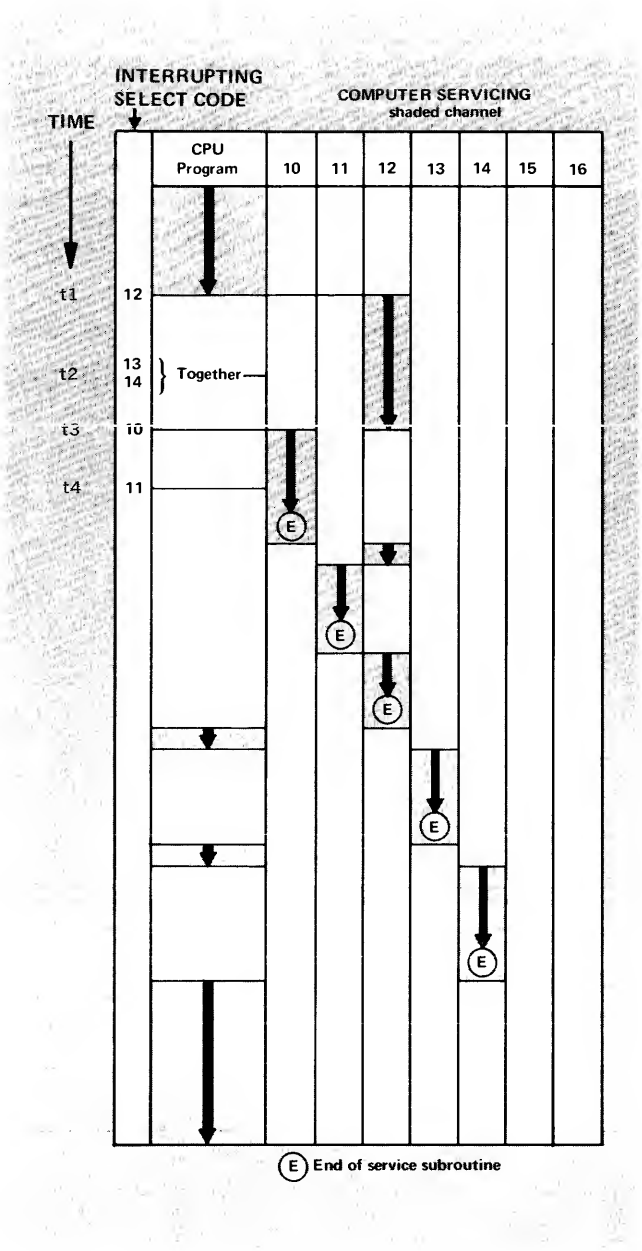


Figure 11. Interrupt Sequences

The routine for channel 12 is not completed when several other devices request service (set flag). First, channels 13 and 14 request simultaneously at t2; however neither has priority over channel 12, so their flags are ignored and channel 12 continues its transfer. But at t3, a higher priority device on channel 10 requests service. This request interrupts the channel 12 transfer and causes the channel 10 transfer to begin. The JSB instruction saves the return address for return to the channel 12 routine.

During the channel 10 transfer, device 11 sets the channel 11 flag(t4). Since it has lower priority than channel 10, device 11 must wait until the end of the channel 10 routine. And since channel 10, when it ends, contains a return address to the channel 12 routine, program control temporarily returns to channel 12 (even though the waiting channel 11 has higher priority). The JMP,I instruction used for the return inhibits all interrupts until fully executed (plus one phase of the next instruction). At the end of this short interval, the channel 11 interrupt request is granted.

When channel 11 has finished its routine, it returns control to channel 12, which at last has sufficient priority to complete its routine. Since channel 12 has been saving a return address in the main CPU program, it returns control to this point.

The two waiting interrupt requests from channels 13 and 14 are now enabled. Since channel 13 has higher priority, it goes first. At the end of its routine, it temporarily returns control to the CPU program. Then the lowest priority channel, 14, interrupts and completes its transfer. Finally, control is returned to the CPU program, which continues processing.

INTERFACE ELEMENTS

The interface card provides a communication link between the computer and an external device. There are three basic elements on the interface card which either the computer or device can control in order to effect the necessary communication. These elements are as follows:

CONTROL BIT

This is a one-bit flip-flop register used by the computer to turn on the device channel. When set, the control bit generates a start command to the device, telling it to begin one operation cycle (e.g., read or write one character or word). The interface cannot interrupt unless the control bit is set. The control bit is set by a STC (set control) instruction and cleared by a CLC (clear control) instruction, with a specific select code (e.g., STC 12 or CLC 12). The device cannot affect the control bit.

FLAG BIT

This is a one-bit flip-flop register primarily used by the device to indicate, when set, that transmission between the device and the interface buffer has been completed. Computer instructions can also set the flag (STF), clear the flag (CLF), test if it is set (SFS), and test if it is clear (SFC). The device cannot clear the flag bit. If the corresponding control bit is set, priority is high, and the interrupt system is enabled, setting the flag bit will cause an interrupt to the location corresponding to the device's select code.

BUFFER

This is a flip-flop register for intermediate storage of data. Typically the data capacity is 8 or 16 bits, but this is entirely dependent on the type of device.

I/O DATA TRANSFER

The preceding paragraphs of this section have discussed the individual features of the I/O system. The following paragraphs show how data is actually transferred under interrupt control. The sequences are highly simplified in order to present an overall view without the involvement of software operating systems and device drivers. For more detailed information refer to the documentation supplied with the appropriate software system or interface kit.

INPUT TRANSFER

The upper part of figure 12 illustrates the sequence of operations for an input transfer. Note that some of the operations are under control of the computer program (programmer's responsibility) and some of the operations are automatic. The sequence is as follows:

The operation begins with a programmed instruction to set control and clear flag on the addressed interface card (1). In this example it is assumed that the interface card is installed in the slot for select code 12; thus the instruction is STC 12,C. Since the next few operations are under automatic control of the hardware, the computer program may continue executing other instructions.

Setting the control bit causes the interface card to issue a start command (2) to the external device. The device then proceeds with its electromechanical process of reading a character. When it has done so, it sends a signal (done) back to the interface card, along with the data character (3).

At the interface card the "done" signal sets the flag bit. The flag, in turn, generates an interrupt (4) — provided the interrupt conditions previously mentioned are met. That is, the interrupt system must be on (STF 00 previously given), no higher priority interrupt may be requesting, and the control bit must be set (done in step 1).

The interrupt causes the current computer program to be suspended, and control is transferred to a service subroutine (5). It is the programmer's responsibility to provide the linkage between the interrupt location (00012 in this case) and the service subroutine. Also, it is the programmer's responsibility to include in his service subroutine the instructions for processing of the data (loading into an accumulator, manipulating if necessary, and storing into memory).

The subroutine may then issue further STC 12,C commands to transfer additional characters. One of the final instructions in the service subroutine must be a clear control (CLC 12 in this case). This step (6) allows lower priority devices to interrupt (equivalent to re-enabling a gate in figure 10) and restores the channel to its static "ready" condition — control cleared and flag set. This condition is initially established by the computer at turn-on, and it is the programmer's responsibility to return the channel to the same condition on completion of each transfer.

At the end of the subroutine, control is returned to the interrupt program via previously established linkages.

OUTPUT TRANSFER

The lower part of figure 12 illustrates the sequence of operations for an output transfer. Again note the distinction between programmed and automatic operations.

It is assumed that the data to be transferred has been loaded into the A-register and is in a form suitable for output. The interface card is assumed to be installed in the slot for select code 13.

The operation begins with a programmed instruction to transfer the data from the A-register to the interface buffer (1). The instruction in this example is OTA 13. This is followed (2) by an instruction to set control and clear flag; i.e., STC 13,C. Since the next few operations are under automatic control of the hardware, the computer program may continue executing other instructions.

Setting the control bit causes the interface card to read out the buffer data to the device and to issue a start command (3). The device proceeds to write the data, and when it has finished the device sends a signal (done) back to the interface card (4).

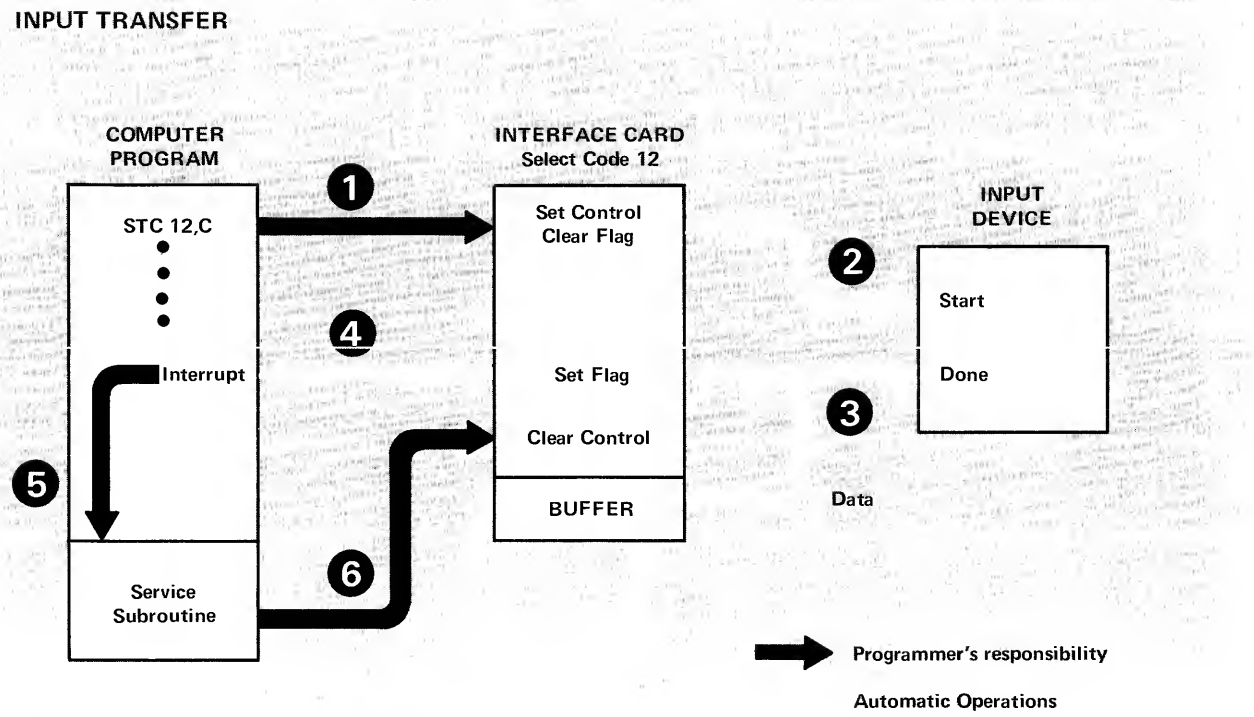
At the interface card the "done" signal sets the flag bit. The flag, in turn, generates an interrupt (5) — provided the interrupt system is on, priority is high, and the control bit is still set (from step 2).

The interrupt causes the current computer program to be suspended, and control is transferred to a service subroutine (6). It is the programmer's responsibility to provide the linkage between the interrupt location (00013 in this case) and the service subroutine. The detailed contents of the subroutine are also the programmer's responsibility, and will vary with the type of device.

The subroutine may then output further data to the interface card and re-issue the STC 13,C command for additional character transfers. One of the final instructions in the service subroutine must be a clear control (CLC 13). This step (7) allows lower priority devices to interrupt, and restores the channel to its static "ready" condition — control cleared and flag set. At the end of the subroutine, control is returned to the interrupted program via previously established linkages.

NON-INTERRUPT TRANSFERS

It is also possible to transfer data without using the interrupt system. This involves a "wait-for-flag" method, in which the computer commands the device to operate and then waits for the completion response. It is therefore



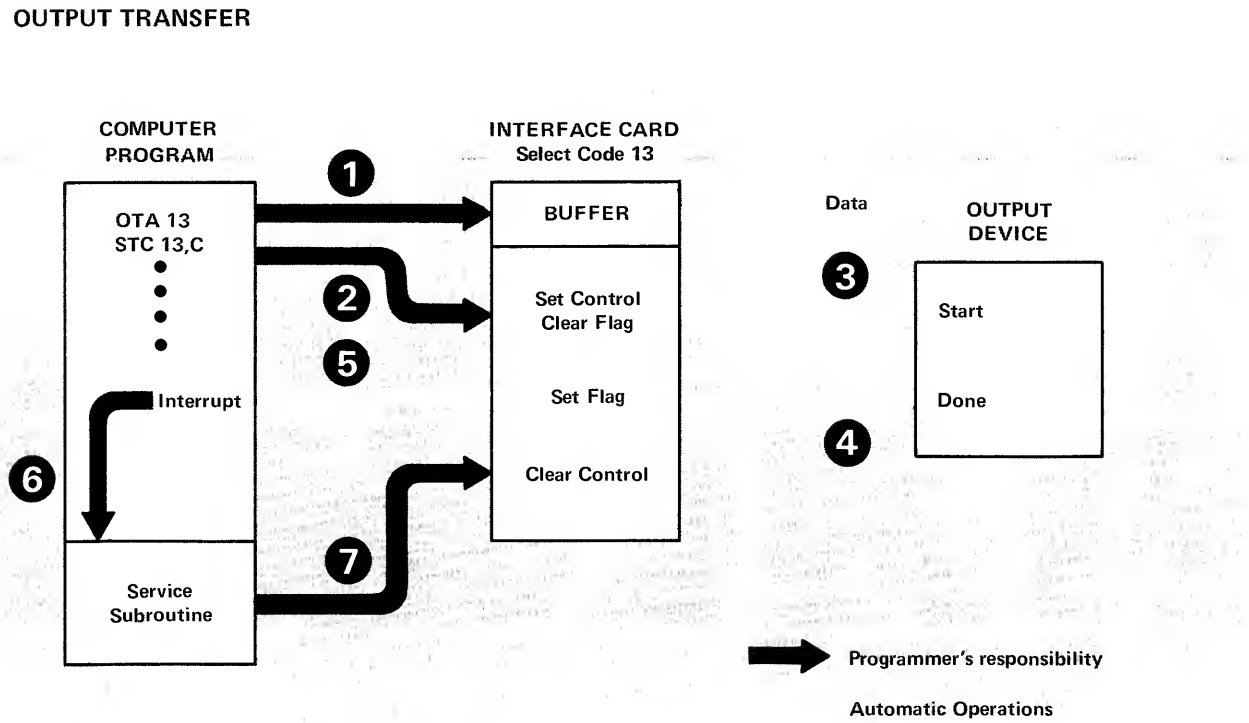


Figure 12. Input/Output Transfers

assumed that computer time is relatively unimportant. The programming is very simple, consisting of only four words of in-line coding, as shown in table 9. Each of these routines will transfer one word or character of data. It is assumed that the interrupt system is turned off (STF 00 not previously given).

INPUT. As before, a STC 12,C instruction begins the operation by commanding the device to read one word or character. The computer then goes into a waiting loop, repeatedly checking the status of the flag bit. If the flag is not set, the JMP *-1 instruction causes a jump back to the SFS instruction. (The *-1 operand is assembler notation for "this location minus one".) When the flag is set, the skip condition for SFS is met and the JMP instruction is skipped. The computer thus exits from the waiting loop, and the LIA 12 instruction loads the device's input data into the A-register.

OUTPUT. The first step of output is to transfer the data to the interface buffer; the OTA 13 instruction does this. Then STC 13,C commands the device to operate and accept the data. The computer then goes into its waiting loop, the same as described in the preceding paragraph. When the flag is received, indicating that the device has accepted the output data, the computer exits from the loop. (The final NOP is for illustration purposes only.)

Table 9. Non-Interrupt Transfer Routines

INPUT

INSTRUCTIONS	COMMENTS
STC 12,C	Start device
SFS 12	Is input ready?
JMP *-1	No, repeat previous instruction
LIA 12	Yes, load input into A-register

OUTPUT

INSTRUCTIONS	COMMENTS
OTA 13	Output A-register to buffer
STC 13,C	Start device
SFS 13	Has device accepted the data?
JMP *-1	No, repeat previous instruction
NOP	Yes, proceed

DIRECT MEMORY ACCESS

As indicated earlier in figure 8, the purpose of the direct memory access (DMA) option is to provide a direct data path, software assignable, between memory and a high-speed peripheral device.

DMA accomplishes this purpose by stealing a memory cycle instead of interrupting to a service subroutine. The DMA option for the 2100A Computer is capable of stealing every consecutive memory cycle, and thus can transfer data at rates up to 1,020,400 words per second.

There are two DMA channels, each of which may be separately assigned to operate with any I/O interface, including those in an HP 2155A I/O Extender. When both DMA channels are in simultaneous operation, channel 1 has priority over channel 2. The combined maximum transfer rate for both channels operating together is 1,020,400 words per second; the rate available to channel 2 is then the rate difference between 1,020,400 and channel 1's actual rate.

When DMA is accessing memory, it has priority over CPU access of memory. Thus the rate available to the CPU when DMA is operating is the difference between 1,020,400 words per second and the actual transfer rate of DMA channels 1 and 2 combined.

DMA transfers are on a full-word basis; hardware packing and unpacking of characters is not provided. The word count register is a full 16 bits in length.

DMA transfers are accomplished in blocks. The transfer is initiated by an initialization routine, and from then on operation is under automatic control of the hardware. The initialization routine tells DMA which direction to transfer the data (in or out), where in memory to put or take data, which I/O channel to use, and how much data to transfer. Completion of the block transfer is signalled by an interrupt to location 00006 (for channel 1) or location 00007 (for channel 2) if the interrupt system is enabled. It is also possible to check for completion by testing the status of the flag for select code 06 or 07, or by interrogating the word count register with an LIA/B to select code 02 (for channel 1) or select code 03 (for channel 2). A block transfer can be aborted with an STF 06 or 07 instruction.

DMA OPERATION

Figure 13 illustrates the sequence of operations for a DMA transfer. Comparison with conventional transfers (figure 12) shows that much more of the operation is automatic. Remember that the procedures in figure 12 must be repeated for each word or character. In figure 13, the automatic DMA operations will transfer a block of data of any size, limited only by the availability of memory space.

The sequence of events is as follows. (Input transfer is illustrated; the minor differences for output are explained in text.)

The initialization routine sets up the control registers on the DMA card (1) and issues the first start command (STC 12,C) directly to the interface card. (If the operation

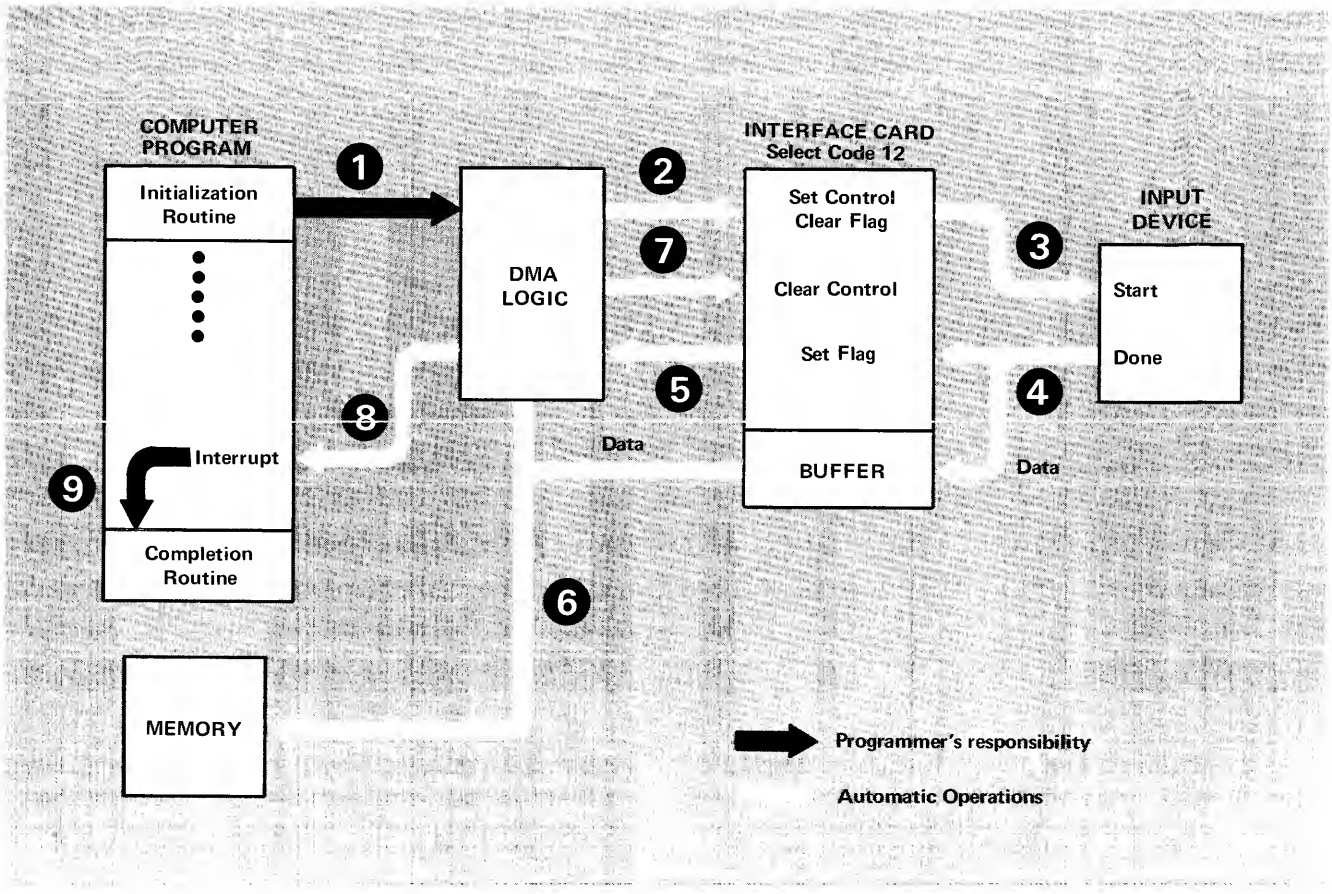


Figure 13. DMA Transfers

is output, the buffer is also loaded at this time.) The DMA option is then turned on and the computer program continues with other instructions.

Setting control and clearing flag on the interface card (2) causes a start command (3) to the external device (with data if output). The device goes through its read or write cycle and returns a "done" signal (4), with data if input. The set flag, regardless of priority, immediately requests DMA to steal a memory cycle (5) and a word is transferred into (or out of) memory (6). The process now repeats back to the beginning of this paragraph to transfer the next word.

After the specified number of words have been transferred, the control bit is cleared (7). Then DMA generates an interrupt (8), and program control is forced to a completion routine (9), the contents of which is the programmer's responsibility.

DMA INITIALIZATION

The information required to initialize DMA (direction, memory allocation, I/O channel assignment, and block length) are given by three control words. These three words must be addressed specifically to the DMA card. Figure 14 shows the format of the three control words.

Control Word 1 (CW1) identifies the I/O channel to be used, and provides for two options, selectable by the programmer as follows.

Bit 15
- 1: give STC (in addition to CLF) to I/O channel at end of each DMA cycle (except on last cycle, if input)
- 0: no STC

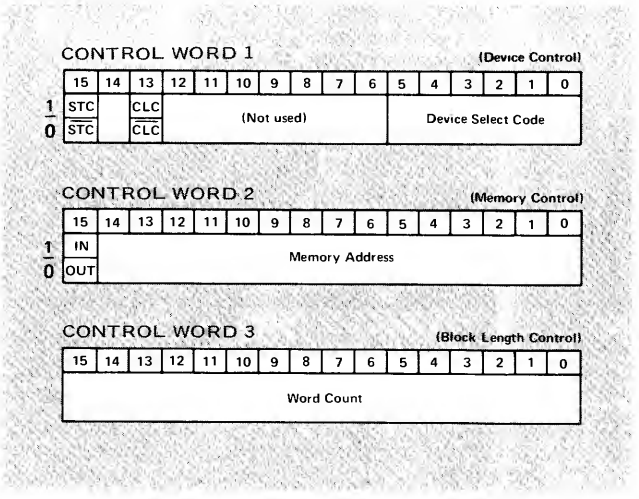


Figure 14. DMA Control Word Formats

Bit 13
- 1: give CLC to I/O channel at end of block transfer
- 0: no CLC

Control Word 2 (CW2) gives the starting memory address for the block transfer, and Bit 15 determines whether data is to go into memory (1) or out of memory (0).

Control Word 3 (CW3) is the 2's complement of the number of words to be transferred into or out of memory; i.e., the length of the block. This number can be from -1 to -32,768, although it is limited in the practical case by available memory.

Table 10 gives the basic program sequence for outputting the control words to DMA. As shown in this table, CLC 2 and STC 2 perform switching functions to prepare the logic for either CW2 or CW3. The device is assumed to be in I/O channel 10, and it is also assumed that its start command is STC 10B,C. The sample values of CW1, CW2, CW3 will read a block of 50 words and store these in locations 200 through 261 (octal). STC 6,C starts the DMA operation. A flag-status method for detecting end-of-transfer is used in this example; an interrupt to location 00006 could be substituted for this test.

The program in table 10 could easily be changed to operate on channel 2 by changing select codes 2 to 3, and 6 to 7.

One important difference should be noted when doing a DMA input operation from a disc or drum. Due to the asynchronous nature of disc or drum memories and the design of the interface, the order of starting must be reversed from the order given; i.e., start DMA first, then the disc.

Table 10. Program to Initialize DMA

LABEL	OPCODE	OPERAND	COMMENTS
ASGN1	LDA	CW1	Fetches control word 1 (CW1) from memory and loads it in A-register.
	OTA	6	Outputs CW1 to DMA Channel 1.
MAR1	CLC	2	Prepares Memory Address Register to receive control word 2 (CW2).
	LDA	CW2	Fetches CW2 from memory and loads it in A-register.
	OTA	2	Outputs CW2 to DMA Channel 1.
WCR1	STC	2	Prepares Word Count Register to receive control word 3 (CW3).
	LDA	CW3	Fetches CW3 from memory and loads it in A-register.
	OTA	2	Outputs CW3 to DMA Channel 1.
STRT1	STC	10B,C	Start input device.
	STC	6B,C	Activate DMA Channel 1.
	SFS	6	Wait while data transfer takes place or, if interrupt processing is
	JMP	*-1	used, continue program.
.	.	.	
.	.	.	
.	.	.	
	HLT		Halt
CW1	OCT	120010	Assignment for DMA Channel 1 (ASGN1); specifies I/O Channel select code address (10_8), STC after each word is transferred, and CLC after final word is transferred.
CW2	OCT	100200	Memory Address Register control. DMA Channel 1 (MAR1); specifies memory input operation and starting memory address (200_8).
CW3	DEC	-50	Word Count Register control. DMA Channel 1 (WCR1); specifies the 2's complement of the number of character words in the block of data to be transferred (50_{10}).

OPERATING CONTROLS AND INDICATORS

SECTION V

The front panel of the 2100A Computer is available in two configurations: an operator panel (standard) and a controller panel (optional).

The operator panel provides display and control of the working registers, phase status and fault indicators, and operating controls.

The controller panel may be used in applications where an operator panel is seldom required. The panels are easily interchangeable so that, if desired, installations having more than one 2100A Computer may share an operator panel among several units.

This section describes the functions of the controls and indicators on both versions of the panel, plus basic operating procedures.

OPERATOR PANEL

Figure 15 illustrates the operator panel and briefly describes the function of each control and indicator. The following paragraphs provide additional explanatory information. Functions are grouped according to the type of operation.

16-BIT REGISTERS

The DISPLAY REGISTER displays the contents of any one of the six 16-bit working registers when in the halt mode. (Only the S-register is displayed in the run mode.) An illuminated bit pushbutton is a "1"; a non-illuminated bit pushbutton is a "0". The bit content changes state each time the pushbutton is pressed, and the entire display may be cleared by pressing CLEAR DISPLAY.

When power is initially turned on, the S-register is automatically selected. Thereafter, while in halt mode, any of the six registers may be selected by pressing the appropriate select switch: A, B, P, M, S, or MEMORY DATA. The register currently selected for display is indicated by lighting of the pushbutton.

After a programmed or manual halt, MEMORY DATA is automatically selected. This causes the contents of the last accessed memory cell to be displayed — which will be the halt instruction code in the case of programmed halts.

As long as a register is being displayed, the original contents of that register may be redisplayed, if altered by pressing DISPLAY REGISTER pushbuttons, simply by pressing the same select pushbutton again (A, B, P, M, S, or MEMORY DATA). However, when any other select pushbutton is pressed (or if the computer is run or stepped) the last indicated display becomes the new contents for that register, and the old contents are lost.

Note that pressing the M pushbutton displays the address of a memory location, and pressing MEMORY DATA displays the contents of that location. Depending on which of these is selected, consecutive addresses or consecutive contents for adjacent memory cells (either higher or lower) may be displayed by repetitively pressing INCREMENT M or DECREMENT M. These two pushbuttons are only momentarily illuminated when pressed. Pressing the P pushbutton also sets the fetch phase, so that execution may begin (at the location indicated by the P-register) simply by pressing RUN.

FAULT INDICATORS

Provision is made to indicate two possible hardware faults. One is a parity error as a result of reading from memory. If the PARITY light is on, a parity error has occurred. In the halt mode, the light may be turned off by pressing INTERNAL PRESET. In the run mode, the light is turned off by a parity error interrupt, and thus is not ordinarily on long enough to be visible.

The other indicated hardware fault is power failure. If the ARS/$\overline{\text{ARS}}$ switch is set to ARS (auto-restart) and location 04 contains a HLT instruction, the EXTERNAL PRESET pushbutton will light on restoration of power, and the machine will halt. The light is turned off by pressing the EXTERNAL PRESET pushbutton. (In a restart routine the light would be turned off by the CLC 04 instruction.)

PHASE STATUS INDICATORS

There are three indicators which signal the state of the computer: FETCH, IND (for indirect), and EXECUTE. The next phase to occur if the computer is run or stepped is the phase indicated by the lighted status indicator. Thus if the FETCH light is on, the computer will fetch an instruction from the address currently pointed to by the P-register when the computer is run or stepped. (It should be noted that indirect references for the extended arithmetic instructions are obtained in an Execute phase, not an Indirect phase.) The indicators are also operative in the run mode.

1-BIT REGISTERS

The contents of the Extend and Overflow registers are continuously displayed by the EXTEND and OVF pushbutton lights (in both halt and run modes). If the pushbutton light is on, the register content is a "1"; if not on, the register content is a "0". In the halt mode, the content changes state each time the pushbutton is pressed.

OPERATING CONTROLS

The eight pushbuttons grouped together as operating controls generally control start/stop and other related functions. Since the effects of each pushbutton differ one from another, they are discussed separately below.

INTERRUPT SYSTEM. This pushbutton indicates and controls the state of the interrupt system. When the pushbutton light is on, the interrupt system is enabled (flag set). When the light is off, the interrupt system is disabled (flag clear). Each time the pushbutton is pressed, while the computer is halted, the flag changes state.

INSTR STEP. This pushbutton is used to advance program execution by instruction. The program advances one instruction each time the pushbutton is pressed. If the RUN light stays on, an infinite indirect loop is indicated; press HALT to terminate the loop.

EXTERNAL PRESET. This pushbutton disables the input/output channels. From I/O address 06 and up, all Control flip-flops are cleared and flag flip-flops are set. If the EXTERNAL PRESET pushbutton lights, a power failure has occurred (see description under Fault Indicators).

INTERNAL PRESET. This pushbutton presets the computer to the fetch phase, clears the PARITY indicator, clears overflow, and disables both the interrupt system and the memory protect logic.

HALT/CYCLE. In the run mode, this pushbutton is used to halt the computer at the end of the current phase. The pushbutton lights when the computer halts, and all other panel controls become enabled. In the halt mode, the pushbutton may be used to advance program execution by phase. One phase occurs (and the light goes off momentarily) each time the pushbutton is pressed.

LOADER ENABLE. This pushbutton enables access to the basic binary loader (last 64 locations of memory) for the purpose of loading binary programs. When the pushbutton is pressed the light goes on, and stays on as long as the loader is enabled. After a programmed or manual halt, the light goes off and the loader is again disabled. (The loader can also be disabled by pressing the pushbutton again.)

RUN. Pressing RUN starts the computer in the current state. The RUN pushbutton light is on while the computer is in the run mode, and all panel controls are disabled except HALT/CYCLE, DISPLAY REGISTER, and CLEAR DISPLAY. Pressing RUN automatically causes the S-register contents to be displayed, and no other register may be selected while the computer is in the run mode. Thus, to the operator, the DISPLAY REGISTER effectively becomes the S-register. This register may be addressed as select code 01 by programmed instructions, and may be manually altered by the operator.

POWER OFF/POWER ON/LOCK ON. This is a three-position key-operated switch, controlling primary power to the computer. The key is removable only in the horizontal POWER OFF and LOCK ON positions. In the LOCK ON position the panel controls are disabled. In the vertical POWER ON position the panel controls are enabled and the key may not be removed.

If it is desired to inhibit the operation of the automatic restart logic when turning power on, the EXTERNAL PRESET pushbutton may be held depressed while turning the power switch.

CONTROLLER PANEL

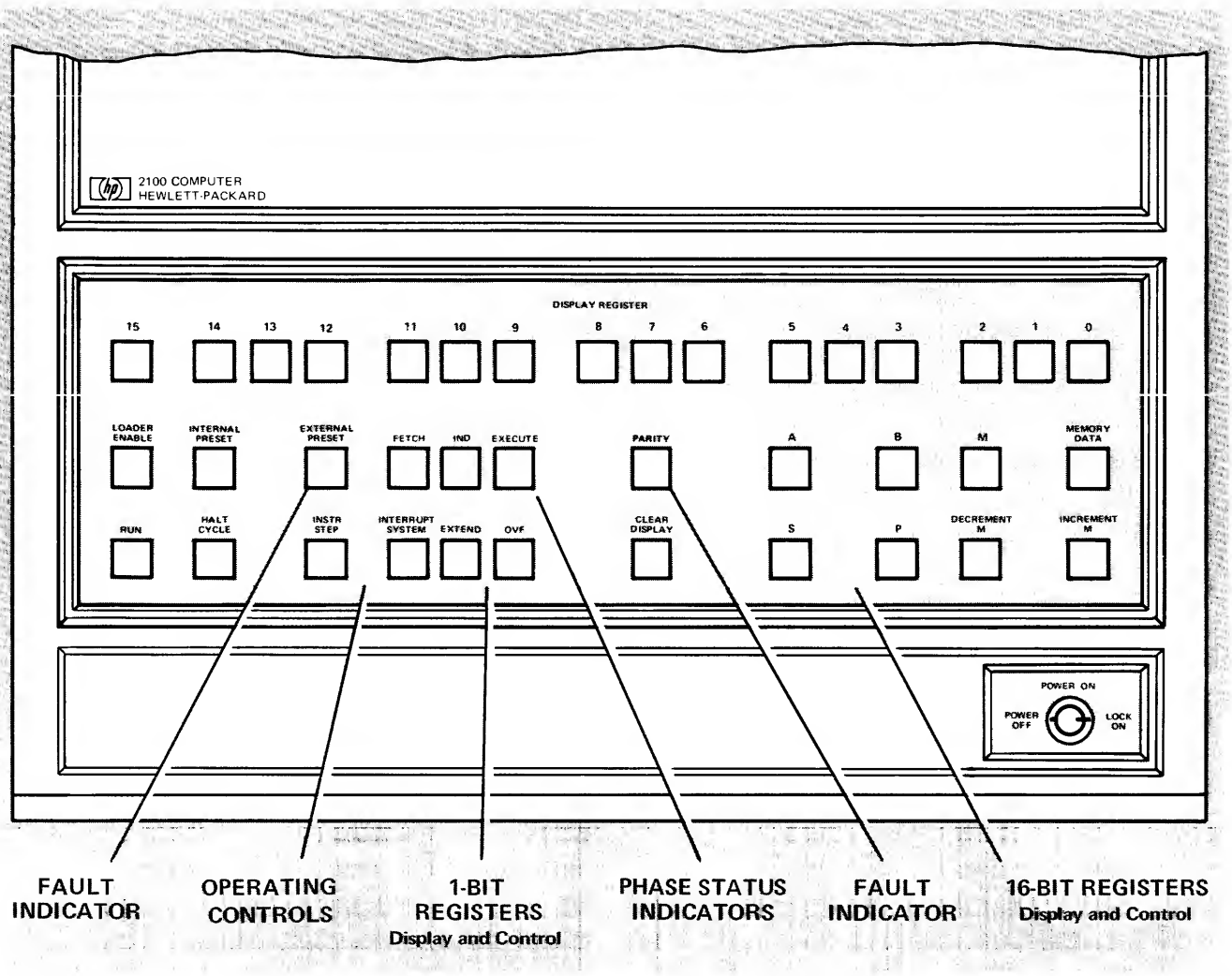
Figure 16 illustrates the optional controller panel and briefly describes the function of each control and indicator. The following paragraphs provide additional explanatory information.

PARITY. If the PARITY light is on, a parity error has occurred as a result of reading from memory. In the halt mode, the light may be turned off by pressing the PRESET pushbutton. In the run mode, the light is turned off by a parity error interrupt.

RUN. Pressing RUN starts the computer in the current state. The RUN pushbutton light is on while the computer is in the run mode, and the PRESET pushbutton is disabled.

HALT. This pushbutton is used to halt the computer at the end of the current phase. The pushbutton lights when the computer halts, and the PRESET pushbutton becomes enabled.

PRESET. This pushbutton disables the input/output channels (clears Control flip-flops and sets flag flip-flops from I/O address 06 and up), turns off the interrupt system, clears the Overflow, A-, B-, and P-registers, clears the PARITY indicator, disables the memory protect logic, and presets the computer to the fetch phase. Pressing the PRESET pushbutton also clears a power failure indication



16-BIT REGISTERS

DISPLAY REGISTER. Bit light on = 1, off = 0. Press switch to complement any bit.

MEMORY DATA. Press to display contents of location referenced by M. Lit when selected. Can press again to redisplay unmodified contents. Automatically selected when computer is halted.

INCREMENT M. Press to increment M. If memory data selected, display is updated.

M. Press to display M-register. Can press again to redisplay unmodified contents.

P. Press to display P-register and set Fetch phase. Can press again to redisplay unmodified contents.

B. Press to display B-register. Can press again to redisplay unmodified contents.

A. Press to display A-register. Can press again to redisplay unmodified contents.

S. Press to display S-register. Can press again to redisplay unmodified contents. Automatically selected in run mode.

CLEAR DISPLAY. Press to clear display register.

1-BIT REGISTERS

OVF. Overflow register. Light on = 1, off = 0. Press to complement.

EXTEND. Extend register. Light on = 1, off = 0. Press to complement.

OPERATING CONTROLS

INTERRUPT SYSTEM. Light on indicates interrupt system enabled. Press to complement.

INSTR STEP. Press to execute single instruction.

EXTERNAL PRESET. Press to clear I/O channels.

INTERNAL PRESET. Set Fetch phase, clear parity error indication and overflow, disable interrupt system and memory protect.

HALT/CYCLE. Halt computer or perform one instruction phase.

LOADER ENABLE. Press to enable/disable loader.

RUN. Start execution, disable panel.

POWER OFF/POWER ON/LOCK ON. Key-operated power switch. Panel disabled in LOCK ON position.

PHASE STATUS INDICATORS

FETCH. Indicates Fetch phase is next.

IND. Indicates Indirect phase is next.

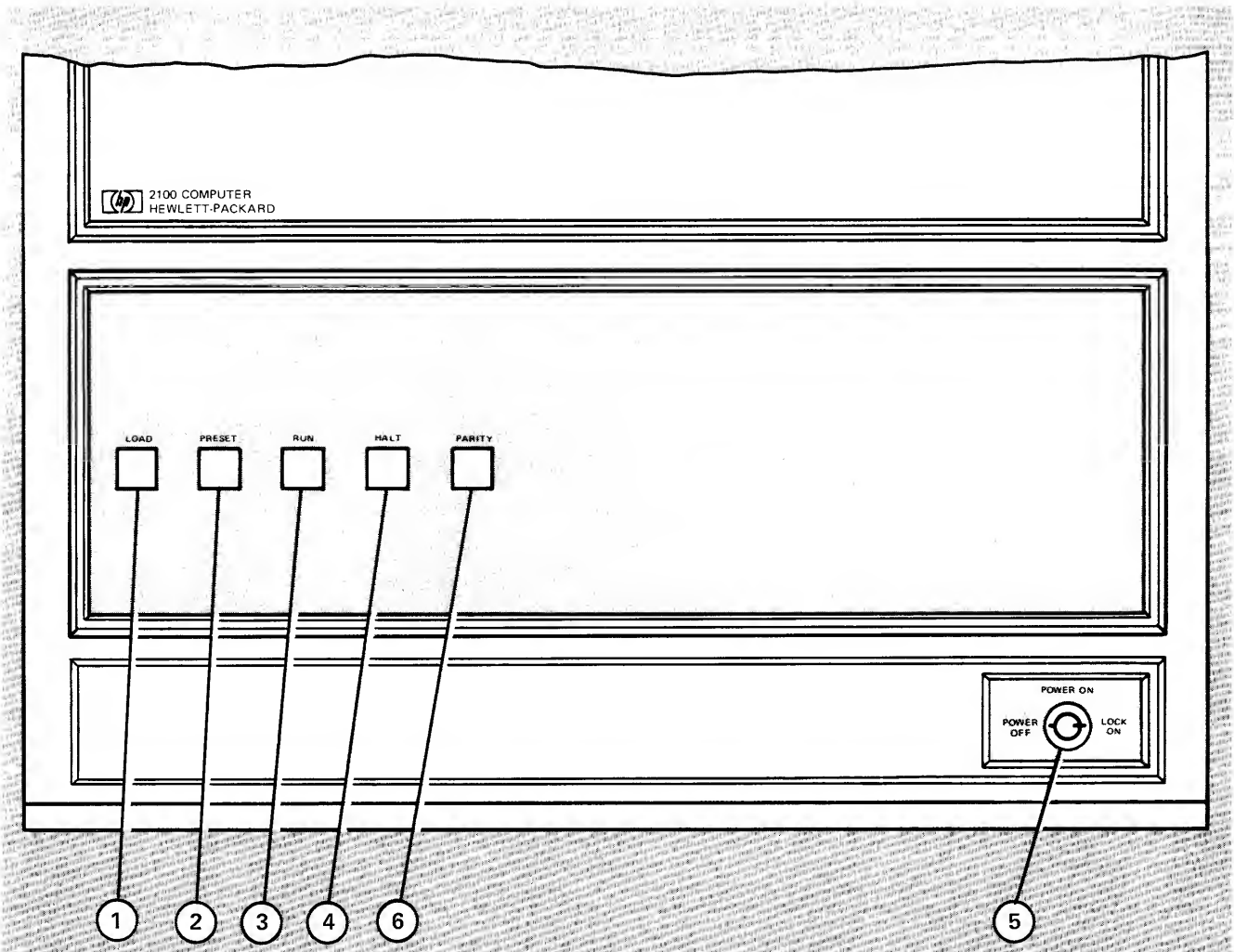
EXECUTE. Indicates Execute phase is next.

FAULT INDICATORS

PARITY. Light on indicates that a memory parity error has occurred (if P.E. HALT mode selected).

EXTERNAL PRESET. Light on indicates a power failure occurred (if location 04 contains HLT).

Figure 15. Operator Panel Controls and Indicators



OPERATING CONTROLS

1. **LOAD.** After preset, press to load program. Light on during load.
2. **PRESET.** Press to set Fetch phase, turn off I/O channels, interrupt system, memory protect, and indications for parity error and power fail. Also clears A-, B-, and P-registers.
3. **RUN.** Press to start program execution. Light on in run mode.
4. **HALT.** Press to halt execution at end of current phase. Light on when halted.
5. **POWER OFF/POWER ON/LOCK ON.** Key-operated power switch. Panel disabled in LOCK ON position.

FAULT INDICATORS

2. **PRESET.** Light on indicates power failure occurred. (Refer to text.)
6. **PARITY.** Light on indicates a memory parity error has occurred, with P.E. INT/HALT switch set to HALT.

Figure 16. Controller Panel Controls and Indicators

(PRESET pushbutton light on) if power has failed and is restored. Note that PRESET will light only if the internal ARS/$\overline{\text{ARS}}$ switch is set to ARS and location 04 contains a HLT instruction.

If the RUN pushbutton is pressed after PRESET, the computer will begin program execution from location 0 P-register = 0). The first two instructions executed will be NOP's (A- and B-registers = 0), and the computer will then begin executing at location 00002. This provides a convenient cold-start linkage in the absence of an operator panel.

LOAD. This pushbutton is used to load a program from a tape reader or disc. In the halt mode, pressing the LOAD pushbutton causes the loader starting address to be loaded into the P-register, enables the loader locations, and starts the run mode. The pushbutton light remains on until a programmed or manual halt occurs. The halt disables the loader and turns off the light.

POWER OFF/POWER ON/LOCK ON. The power control switch is not replaced when panels are interchanged. Refer to the description given previously.

INTERNAL SWITCHES

Although most of the internal switches are intended for checkout or maintenance purposes, two of these are of interest to the user. The following paragraphs describe the functions of these switches. Access to the switches is obtained by removing the computer top cover; each switch is mounted near the top edge of a printed-circuit card, the location of which is specified in the following text.

ARS/$\overline{\text{ARS}}$

The ARS/$\overline{\text{ARS}}$ switch is used to specify the action which the computer should take on recovery from a power failure. With the switch in the ARS position, the computer will interrupt to location 00004 when power returns to normal operating levels; this permits entry to a restart program. With the switch in the $\overline{\text{ARS}}$ position, the computer will halt on recovery of power. The ARS/$\overline{\text{ARS}}$ switch is located on the I/O control card in slot 7.

INT/HALT

The P.E. INT/HALT switch is used to specify the action which the computer should take on detection of a memory parity error. With the switch in the INT position, the computer will interrupt to location 00005 for entry to a parity error subroutine. With the switch in the HALT position, the computer will halt. The P.E. INT/HALT switch is located on the I/O buffer card in slot 8.

OPERATING PROCEDURES FOR OPERATOR PANEL

The following procedures describe, in general, the basic load and run operations for the 2100A Computer. Depending on whether or not a disc is present in the system, loading is accomplished by means of the basic binary loader (BBL) or basic binary disc loader (BBDL). All procedures require that the power-switch key be in the vertical POWER ON position (panel enabled).

LOADING WITH BASIC BINARY LOADER

It is assumed that the basic binary loader program is present in memory, and is properly configured for the channel number of the input device and for the size of memory. Refer to the software operating manual for the procedure required to configure the loader. Loading is accomplished as follows:

a. Turn on the input device and prepare for reading (e.g., load tape in tape reader). The input program must be in binary form, containing absolute addresses.

b. Press S to select the S-register. This will cause the S-register contents to be displayed in the DISPLAY REGISTER.

c. Clear bits 0 and 15 of the display. (These bits are to be set only for certain nonloading check operations; refer to software operating manual.) The status of the remaining bits is not significant.

d. Press P to select the P-register. This will cause the P-register contents to be displayed in the DISPLAY REGISTER.

e. Set the display to the starting address of the basic binary loader, according to table 11.

f. Press EXTERNAL PRESET and INTERNAL PRESET. This initializes the external hardware (I/O channels) and the internal hardware (central processor).

g. Press LOADER ENABLE, and then press RUN. The lights for both switches will remain on while the input operation is in progress.

h. When the input device stops, the HALT light will go on, RUN and LOADER ENABLE lights will go off, and the DISPLAY REGISTER should indicate 102077 (octal), with MEMORY DATA automatically selected. The load is complete.

If the halt code is not 102077 when the device stops, there has been an error in the loading process. Two possible error conditions are indicated by the loader, which changes the halt code to identify the type of error. A halt code of

Table 11. Loader Starting Addresses

MEMORY SIZE	STARTING ADDRESS OF LOADER	
	For Paper Tape	For Disc
4K	07700	
8K	17700	17760
12K	27700	27760
16K	37700	37760
24K	57700	57760
32K	77700	77760

102055 indicates an address error; check if the proper tape is being read, or if it is in backwards. A halt code of 102011 indicates a checksum error; check for possible bad tape, or dirty tape reader or tape.

LOADING WITH DISC LOADER

If a disc is present in the system, the basic binary disc loader (rather than the basic binary loader) occupies the protected loader locations. This loader allows loading from either disc or paper tape. The choice is made by selecting one of two possible starting addresses, as indicated in table 11. For paper tapes the procedure is the same as described above for the basic binary loader; steps "b" and "c" can be omitted.

The following procedures for disc loading assume that the basic binary disc loader is present in memory, and is properly configured for the I/O channel numbers being used and for the size of memory. The input program on disc must be in binary form, containing absolute addresses.

a. Press P to select the P-register. This will cause the P-register contents to be displayed in the DISPLAY REGISTER.

b. Set the display to the starting address in the loader which is appropriate to the input source (disc) and memory size, as indicated in table 11.

c. Press EXTERNAL PRESET and INTERNAL PRESET. This initializes the external hardware (I/O channels) and the internal hardware (central processor).

d. Press LOADER ENABLE, and then press RUN.

In the case of disc loading, the load may occur too quickly to detect visually from the panel lights. However, a correct load is indicated (for either tape or disc) by a display of 102077 (octal), with MEMORY DATA automatically selected. (The P-register contents could also be checked. With tape loading, the address should have changed from the first to the last address, plus one, of the loader. With disc loading, the P-register should contain octal 10.)

If the displayed halt code is not 102077 when the load is complete, there has been an error. For disc loading, the error indications are undefinable. For paper tape loading, the loader will alter the halt code to identify the type of error, as described above for basic binary loader operation.

MANUAL LOADING

Short programs may also be loaded manually from the front panel.

a. Press M to select the M-register. This will cause the M-register contents to be displayed in the DISPLAY REGISTER.

b. Set the display to indicate the desired starting address for the program.

c. Press MEMORY DATA. This will cause the current contents of the memory location to be displayed in the DISPLAY REGISTER.

d. Change the displayed contents to the binary instruction code for the first instruction of the program to be loaded. (It may be faster to press CLEAR DISPLAY and begin coding from an all-zero display.)

e. Press INCREMENT M. The contents of the next memory location will be displayed, and the M-register, although not displayed, will be incremented.

f. Enter the next instruction into the DISPLAY REGISTER.

g. Repeat steps "e" and "f" until the entire program has been loaded. To check which location is being displayed, M can be pressed at any time in the procedure to display the current address.

RUNNING PROGRAMS

To run a program after it has been loaded:

a. Press P to select the P-register.

b. Set the display to the starting address of the program.

c. Press EXTERNAL PRESET AND INTERNAL PRESET.

d. Press RUN.

The RUN light will be on as long as the program is running. All panel controls except HALT/CYCLE, DISPLAY REGISTER, and CLEAR DISPLAY are disabled. The S-register is automatically selected, and may be manually changed via the DISPLAY REGISTER.

Additionally, if desired, the display and halt controls may also be disabled by turning the power-switch key to the horizontal LOCK ON position. The key may be removed in this position, and thus protect the state of the computer from accidental tampering.

OPERATING PROCEDURES FOR CONTROLLER PANEL

LOADING PROGRAMS

It is assumed that the loader program is present in memory, and that the loader and the panel are properly configured for the type of loader (paper tape or disc), the channel number of the input device, and the applicable memory size. Refer to the 2100A Installation and Maintenance manual for the procedure required to configure the panel, and to the software operating manual for the procedure required to configure the loader. Loading is accomplished as follows:

a. Turn on the input device and prepare for reading (e.g., load tape in tape reader). The input program must be in binary form, containing absolute addresses.

b. Press PRESET. This initializes both the external hardware (I/O channels) and the internal hardware (central processor).

c. Press LOAD. The LOAD light will go on and will remain on during the load (or until the pushbutton is released in the case of disc loading). No error checking is provided.

RUNNING PROGRAMS

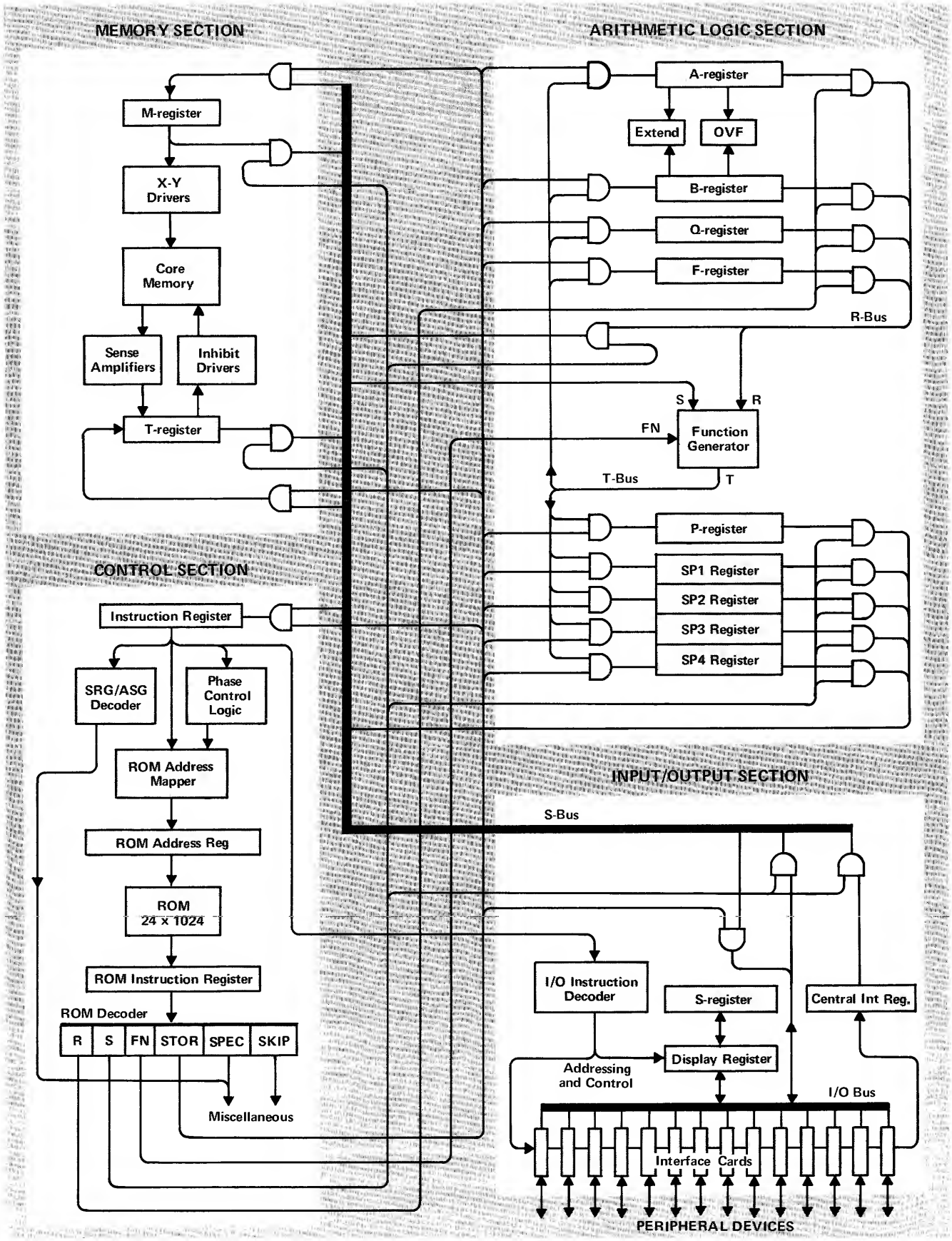
To run the loaded program, press PRESET and then press RUN.

The PRESET switch causes the A-, B-, and P-registers to be cleared, thus causing execution to begin at location 00000 (A-register). The computer executes the NOP instruction contained in the A-register (all-zero word), and also the NOP in the B-register. Then, in location 00002, a JMP instruction causes a jump to the starting instruction of the program.

The RUN light will be on as long as the program is running. Only the HALT switch is enabled. However, even this switch may be disabled by turning the power-switch key to the horizontal LOCK ON position. The key may be removed in this position, and thus protect the state of the computer from accidental tampering.

APPENDIX

FUNCTIONAL BLOCK DIAGRAM



PROCESSOR LOGIC ELEMENTS

MEMORY SECTION

M-Register. Contains binary address of memory cell being accessed. Contents gated to or from S-bus by S and STOR fields (respectively) of ROM instruction word.

X-Y Drivers. Current drivers which strobe all 17 cores in a given memory location, one direction for reading, the opposite direction for writing.

Core Memory. Array of magnetic cores for data storage. Magnetization direction of each core indicates "1" bit or "0" bit. (17 bits per location.)

Sense Amplifiers. Pulse amplifiers to detect which of the 17 cores change state when reading the contents of one location. Resulting signals cause duplication of the bit pattern into the T-register. (17th bit goes to parity checking logic, not shown.)

Inhibit Drivers. Current drivers which prevent certain cores from changing state (according to the bit pattern in the T-register) when writing into memory. Causes duplication of T-register contents into the memory location.

T-register. 16-bit register to receive data from memory, and hold data for storage into memory. Contents can be gated to or from S-bus by ROM S or STOR fields.

CONTROL SECTION

Instruction Register. 16-bit register to receive instruction word from T-register in fetch phase. Loaded from S-bus by ROM STOR field.

SRG/ASG Decoder. Register reference instructions are partially decoded separate from ROM. Resulting control signals directly affect A-, B-, or P-registers.

Phase Control Logic. Causes ROM address mapper to set up a ROM address corresponding to the current instruction phase.

ROM Address Mapper. Uses the instruction register code to find the ROM starting address for an instruction, and the address for each phase of that instruction.

ROM Address Register. Contains the binary address of the ROM location being read out.

ROM. (Read-only memory.) A matrix of permanently stored instruction codes, addressable to read out any stored code on command. (24 bits per location.)

ROM Instruction Register. 24-bit register to receive ROM instruction words.

ROM Decoder. Decodes ROM instruction codes into control signals, to select which register to read onto the R- and S-buses, as well as where to store S-bus data; also numerous other functions.

ARITHMETIC LOGIC SECTION

A-register. 16-bit accumulator. Loaded from T-bus by ROM STOR field, read to R-bus by R field.

B-register. Second accumulator, same as A-register.

Extend. One-bit register used to extend the A- or B- register to 17 bits. Can also be used independently.

OVF. (Overflow.) One-bit register used to signify an arithmetic overflow due to arithmetic operations with the A- or B-registers. Can also be used independently.

Q-register. 16-bit left-shifting register, used to accumulate quotient in arithmetic division. Not externally accessible.

F-register. Same as Q-register, except accumulates division remainder.

R-bus. 16-bit data bus, one of two data inputs to the function generator. ROM R field reads 1 of 4 registers onto this bus. Can be gated to S-bus by S field.

Function Generator. Performs a specified function (FN) on one or both of the R- and S-bus inputs, and puts the result onto the T-bus. Functions include: addition, subtraction, boolean operations, increment, decrement, etc.

T-Bus. 16-bit data bus to transfer data modified by the function generator to any of nine registers.

P-register. 16-bit register used to hold the address of the current program instruction.

SP(1-4) Registers. 16-bit temporary storage registers used by ROM only.

INPUT/OUTPUT SECTION

I/O Instruction Decoder. Input/output instructions are partially decoded separate from ROM. Resulting signals provide addressing and control functions to the I/O system.

S-register. 16-bit data register. Can be loaded via display register in halt mode. In run mode, S-register is locked to display register; is addressable by select code 01.

Display Register. In halt mode, provides manual loading facility for other registers. In run mode, may be gated via I/O bus to or from S-bus, using select code 01 with S and STOR fields.

Central Interrupt Register. Six-bit register, holds the address of the most recently interrupting function or device.

I/O Bus. 16-bit data bus accessible to all I/O interface cards. Can be gated to or from the S-bus by ROM S and STOR fields.

Interface Cards. One card per I/O channel, allows direct cable connection of peripheral devices to the input/output section of the computer.

OCTAL ARITHMETIC

ADDITION

TABLE

0	01	02	03	04	05	06	07
1	02	03	04	05	06	07	10
2	03	04	05	06	07	10	11
3	04	05	06	07	10	11	12
4	05	06	07	10	11	12	13
5	06	07	10	11	12	13	14
6	07	10	11	12	13	14	15
7	10	11	12	13	14	15	16

EXAMPLE

Add:

```
  3677   octal
+ 1331   octal
 (111-)  carries
 5230    octal
```

MULTIPLICATION

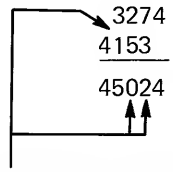
TABLE

1	02	03	04	05	06	07
2	04	06	10	12	14	16
3	06	11	14	17	22	25
4	10	14	20	24	30	34
5	12	17	24	31	36	43
6	14	44	30	36	44	52
7	16	35	34	43	52	61

EXAMPLE

Multiply:

```
  657   octal
X  54   octal
 3274
 4153
45024   octal
```



(Reminder: add in octal)

COMPLEMENT

To find the two's complement form of an octal number. (Same procedure whether converting from positive to negative or negative to positive.)

RULE

1. Subtract from the maximum representable octal value.
2. Add one.

EXAMPLE

Two's complement of 556_8:

```
 177777
-000556
 177221
    +1
 177222
```

177222_8

OCTAL/DECIMAL CONVERSIONS

OCTAL TO DECIMAL

TABLE

OCTAL	DECIMAL
0- 7	0- 7
10-17	8-15
20-27	16-23
30-37	24-31
40-47	32-39
50-57	40-47
60-67	48-55
70-77	56-63
100	64
200	128
400	256
1000	512
2000	1024
4000	2048
10000	4096
20000	8192
40000	16384
77777	32767

EXAMPLE

Convert 463_8 to a decimal integer.

$400_8 = 256_{10}$

$60_8 = 48_{10}$

$3_8 = 3_{10}$

307 decimal

DECIMAL TO OCTAL

TABLE

DECIMAL	OCTAL
1	1
10	12
20	24
40	50
100	144
200	310
500	764
1000	1750
2000	3720
5000	11610
10000	23420
20000	47040
32767	77777

EXAMPLE

Convert 5229_{10} to an octal integer.

$5000_{10} = 11610_8$

$200_{10} = 310_8$

$20_{10} = 24_8$

$9_{10} = 11_8$

12155_8

(Reminder: add in octal)

NEGATIVE DECIMAL TO TWO'S COMPLEMENT OCTAL

TABLE

DECIMAL	2's COMP
-1	177777
-10	177766
-20	177754
-40	177730
-100	77634
-200	177470
-500	177014
-1000	176030
-2000	174040
-5000	166170
-10000	154360
-20000	130740
-32768	100000

EXAMPLE

Convert -629_{10} to two's complement octal.

$-500_{10} = 177014_8$

$-100_{10} = 177634_8$

$-20_{10} = 177754_8$ (Add in octal)

$-9_{10} = 177767_8$

176613_8

For reverse conversion (two's complement octal to negative decimal):

1. Complement, using procedure on facing page.
2. Convert to decimal, using OCTAL TO DECIMAL table.

MATHEMATICAL EQUIVALENTS

$2^{\pm n}$ IN DECIMAL

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.01562 5
128	7	0.00781 25
256	8	0.00390 625
512	9	0.00195 3125
1 024	10	0.00097 65625
2 048	11	0.00048 82812 5
4 096	12	0.00024 41406 25
8 192	13	0.00012 20703 125
16 384	14	0.00006 10351 5625
32 768	15	0.00003 05175 78125
65 536	16	0.00001 52587 89062 5
131 072	17	0.00000 76293 94531 25
262 144	18	0.00000 38146 97265 625
524 288	19	0.00000 19073 48632 8125
1 048 576	20	0.00000 09536 74316 40625
2 097 152	21	0.00000 04768 37158 20312 5
4 194 304	22	0.00000 02384 18579 10156 25
8 388 608	23	0.00000 01192 09289 55078 125
16 777 216	24	0.00000 00596 04644 77539 0625
33 554 432	25	0.00000 00298 02322 38769 53125
67 108 864	26	0.00000 00149 01161 19384 76562 5
134 217 728	27	0.00000 00074 50580 59692 38281 25
268 435 456	28	0.00000 00037 25290 29846 19140 625
536 870 912	29	0.00000 00018 62645 14923 09570 3125
1 073 741 824	30	0.00000 00009 31322 57461 54785 15625
2 147 483 648	31	0.00000 00004 65661 28730 77392 57812 5
4 294 967 296	32	0.00000 00002 32830 64365 38696 28906 25

$10^{\pm n}$ IN OCTAL

10^n	n	10^{-n}
1	0	1.000 000 000 000 000 000 00
12	1	0.063 146 314 631 463 146 31
144	2	0.005 075 341 217 270 243 66
1 750	3	0.000 406 111 564 570 651 77
23 420	4	0.000 032 155 613 530 704 15
303 240	5	0.000 002 476 132 610 706 64
3 641 100	6	0.000 000 206 157 364 055 37
46 113 200	7	0.000 000 015 327 745 152 75
575 360 400	8	0.000 000 001 257 143 561 06
7 346 545 000	9	0.000 000 000 104 560 276 41
112 402 762 000	10	0.000 000 000 006 676 337 66
1 351 035 564 000	11	0.000 000 000 000 537 657 77
16 432 451 210 000	12	0.000 000 000 000 043 136 32
221 411 634 520 000	13	0.000 000 000 000 003 411 35
2 657 142 036 440 000	14	0.000 000 000 000 000 264 11
34 327 724 461 500 000	15	0.000 000 000 000 000 022 01
434 157 115 760 200 000	16	0.000 000 000 000 000 001 63
5 432 127 413 542 400 000	17	0.000 000 000 000 000 000 14
67 405 553 164 731 000 000	18	0.000 000 000 000 000 000 01

MATHEMATICAL EQUIVALENTS

2^x IN DECIMAL

x	2^x	x	2^x	x	2^x
0.001	1.00069 33874 62581	0.01	1.00695 55500 56719	0.1	1.07177 34625 36293
0.002	1.00138 72557 11335	0.02	1.01395 94797 90029	0.2	1.14869 83549 97035
0.003	1.00208 16050 79633	0.03	1.02101 21257 07193	0.3	1.23114 44133 44916
0.004	1.00277 64359 01078	0.04	1.02811 38266 56067	0.4	1.31950 79107 72894
0.005	1.00347 17485 09503	0.05	1.03526 49238 41377	0.5	1.41421 35623 73095
0.006	1.00416 75432 38973	0.06	1.04246 57608 41121	0.6	1.51571 65665 10398
0.007	1.00486 38204 23785	0.07	1.04971 66836 23067	0.7	1.62450 47927 12471
0.008	1.00556 05803 98468	0.08	1.05701 80405 61380	0.8	1.74110 11265 92248
0.009	1.00625 78234 97782	0.09	1.06437 01824 53360	0.9	1.86606 59830 73615

$n \log_{10} 2$, $n \log_2 10$ IN DECIMAL

n	$n \log_{10} 2$	$n \log_2 10$	n	$n \log_{10} 2$	$n \log_2 10$
1	0.30102 99957	3.32192 80949	6	1.80617 99740	19.93156 85693
2	0.60205 99913	6.64385 61898	7	2.10720 99696	23.25349 66642
3	0.90308 99870	9.96578 42847	8	2.40823 99653	26.57542 47591
4	1.20411 99827	13.28771 23795	9	2.70926 99610	29.89735 28540
5	1.50514 99783	16.60964 04744	10	3.01029 99566	33.21928 09489

MATHEMATICAL CONSTANTS IN OCTAL SCALE

$\pi = (3.11037\ 552421)_{(8)}$

$\pi^{-1} = (0.24276\ 301556)_{(8)}$

$\sqrt{\pi} = (1.61337\ 611067)_{(8)}$

$\ln \pi = (1.11206\ 404435)_{(8)}$

$\log_2 \pi = (1.51544\ 163223)_{(8)}$

$\sqrt{10} = (3.12305\ 407267)_{(8)}$

$e = (2.55760\ 521305)_{(8)}$

$e^{-1} = (0.27426\ 530661)_{(8)}$

$\sqrt{e} = (1.51411\ 230704)_{(8)}$

$\log_{10} e = (0.33626\ 754251)_{(8)}$

$\log_2 e = (1.34252\ 166245)_{(8)}$

$\log_2 10 = (3.24464\ 741136)_{(8)}$

$\gamma = (0.44742\ 147707)_{(8)}$

$\ln \gamma = -(0.43127\ 233602)_{(8)}$

$\log_2 \gamma = -(0.62573\ 030645)_{(8)}$

$\sqrt{2} = (1.32404\ 746320)_{(8)}$

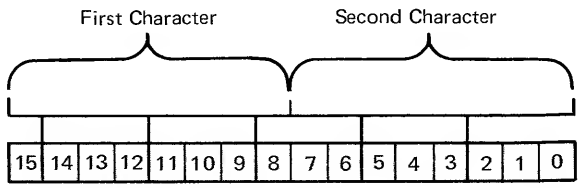
$\ln 2 = (0.54271\ 027760)_{(8)}$

$\ln 10 = (2.23273\ 067355)_{(8)}$

CHARACTER CODES

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
A	040400	000101
B	041000	000102
C	041400	000103
D	042000	000104
E	042400	000105
F	043000	000106
G	043400	000107
H	044000	000110
I	044400	000111
J	045000	000112
K	045400	000113
L	046000	000114
M	046400	000115
N	047000	000116
O	047400	000117
P	050000	000120
Q	050400	000121
R	051000	000122
S	051400	000123
T	052000	000124
U	052400	000125
V	053000	000126
W	053400	000127
X	054000	000130
Y	054400	000131
Z	055000	000132
0	030000	000060
1	030400	000061
2	031000	000062
3	031400	000063
4	032000	000064
5	032400	000065
6	033000	000066
7	033400	000067
8	034000	000070
9	034400	000071
space	020000	000040
!	020400	000041
"	021000	000042
#	021400	000043
$	022000	000044
%	022400	000045
&	023000	000046
'	023400	000047
(	024000	000050
)	024400	000051
*	025000	000052
+	025400	000053
,	026000	000054
–	026400	000055
.	027000	000056
/	027400	000057

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
:	035000	000072
;	035400	000073
<	036000	000074
=	036400	000075
>	037000	000076
?	037400	000077
@	040000	000100
[	055400	000133
\	056000	000134
]	056400	000135
↑	057000	000136
←	057400	000137
ACK	036000	000174
①	036400	000175
ESC	037000	000176
DEL	037400	000177
NULL	000000	000000
SUM	000400	000001
EOA	001000	000002
EOM	001400	000003
EOT	002000	000004
WRU	002400	000005
RU	003000	000006
BELL	003400	000007
FE_0	004000	000010
HT/SK	004400	000011
LF	005000	000012
V_{TAB}	005400	000013
FF	006000	000014
CR	006400	000015
SO	007000	000016
SI	007400	000017
DC_0	010000	000020
DC_1	010400	000021
DC_2	011000	000022
DC_3	011400	000023
DC_4	012000	000024
ERR	012400	000025
SYNC	013000	000026
LEM	013400	000027
S_0	014000	000030
S_1	014400	000031
S_2	015000	000032
S_3	015400	000033
S_4	016000	000034
S_5	016400	000035
S_6	017000	000036
S_7	017400	000037



OCTAL COMBINING TABLES

MEMORY REFERENCE INSTRUCTIONS

Indirect Addressing

Refer to octal instruction codes given on the following page.
To combine code for indirect addressing, merge "100000" with octal instruction code.

REGISTER REFERENCE INSTRUCTIONS

Shift-Rotate Group (SRG)

1. select to operate on A or B
2. select 1 to 4 micros, not more than one from each column.
3. combine octal codes (leading zeros omitted) by inclusive or.
4. order of execution is from column 1 to column 4.

A Operations

1	2	3	4
ALS (1000)	CLE (40)	SLA (10)	ALS (20)
ARS (1100)			ARS (21)
RAL (1200)			RAL (22)
RAR (1300)			RAR (23)
ALR (1400)			ALR (24)
ERA (1500)			ERA (25)
ELA (1600)			ELA (26)
ALF (1700)			ALF (27)

B Operations

1	2	3	4
BLS (5000)	CLE (4040)	SLB (4010)	BLS (4020)
BRS (5100)			BRS (4021)
RBL (5200)			RBL (4022)
RBR (5300)			RBR (4023)
BLR (5400)			BLR (4024)
ERB (5500)			ERB (4025)
ELB (5600)			ELB (4026)
BLF (5700)			BLF (4027)

Alter-Skip Group (ASG)

1. select to operate on A or B.
2. select 1 to 8 micros, not more than one from each column.
3. combine octal codes (leading zeros omitted) by inclusive or.
4. order of execution is from column 1 to column 8.

A Operations

1	2	3	4
CLA (2400)	SEZ (2040)	CLE (2100)	SSA (2020)
CMA (3000)		CME (2200)	
CCA (3400)		CCE (2300)	

5	6	7	8
SLA (2010)	INA (6040)	CLE (6100)	RSS (2001)

B Operations

1	2	3	4
CLB (6400)	SEZ (6040)	CLE (6100)	SSB (6020)
CMB (7000)		CME (6200)	
CCB (7400)		CCE (6300)	

5	6	7	8
SLB (6010)	INB (6004)	SZB (6002)	RSS (6001)

INPUT/OUTPUT INSTRUCTIONS

Clear Flag

Refer to octal instruction codes given on the following page.
To clear flag after execution (instead of holding flag), merge "001000" with octal instruction code.

ASSIGNMENTS OF SELECT CODES

Octal	I/O Address for
00	Interrupt system on (STF) or off (CLF)
01	S-register (for LIA/B, MIA/B) or Overflow register (for STO, CLO, SOS, SOC)
02	DMA channel 1 initialization
03	DMA channel 2 initialization
04	Central interrupt register (for LIA/B, MIA/B)
05	Parity Error option on (STF) or off (CLF); or Memory Protect enable (STC) and violation register (LIA/B)
06	DMA channel 1 on (STC) and status check (SFS or SFC)
07	DMA channel 2 on (STC) and status check (SFS or SFC)
10	Highest priority I/O channel
11 etc.	
25	Last I/O channel if no extender or multiplexer used
25	First I/O channel in extender
26 etc.	
65	Last I/O channel in extender
66 thru 77	Multiplexed I/O available channels

ASSIGNMENTS OF LOW CORE

00000	(A-register)
00001	(B-register)
00002	Not assigned
00003	Not assigned
00004	Power fail interrupt
00005	Parity error interrupt or memory protect interrupt
00006	DMA channel 1 interrupt
00007	DMA channel 2 interrupt
00010 thru 00077	Interrupt locations for 56 I/O device channels, in descending priority

INSTRUCTION CODES IN OCTAL*

Memory Reference

AND	01 (0XX) ---
XOR	02 (0XX) ---
IOR	03 (0XX) ---
JSB	01 (1XX) ---
JMP	02 (1XX) ---
ISZ	03 (1XX) ---
ADA	04 (0XX) ---
ADB	04 (1XX) ---
CPA	05 (0XX) ---
CPB	05 (1XX) ---
LDA	06 (0XX) ---
LDB	06 (1XX) ---
STA	07 (0XX) ---
STB	07 (1XX) ---

↑ Binary

Shift-Rotate

NOP	000000
CLE	000040
SLA	000010
SLB	004010
ALS	001000
BLS	005000
ARS	001100
BRS	005100
RAL	001200
RBL	005200
RAR	001300
RBR	005300
ALR	001400
BLR	005400
ERA	001500
ERB	005500
ELA	001600
ELB	005600
ALF	001700
BLF	005700

Alter-Skip

CLA	002400
CLB	006400
CMA	003000
CMB	007000
CCA	003400
CCB	007400
SEZ	002040
CLE	002100
CME	002200
CCE	002300
SSA	002020
SSB	006020
SLA	002010
SLB	006010
INA	002004
INB	006004
SZA	002002
SZB	006002
RSS	002001

Input/Output

HLT	1020 --
STF	1021 --
CLF	1031 --
SFC	1022 --
SFS	1023 --
MIA	1024 --
MIB	1064 --
LIA	1025 --
LIB	1065 --
OTA	1026 --
OTB	1066 --
STC	1027 --
CLC	1067 --
STO	102101
CLO	103101
SOC	102201
SOS	102301

Extended Arithmetic

MPY	100200
DIV	100400
DLD	104200
DST	104400
ASR	1010 (01X) -
ASL	1000 (01X) -
LSR	1010 (10X) -
LSL	1000 (10X) -
RRR	1011 (00X) -
RRL	1001 (00X) -

↑ Binary

*Assuming: no indirect addressing
no combined microinstructions
shifts taken in first position only
hold flag after I/O execution

Refer to preceding page for octal combining tables

INSTRUCTION CODES IN BINARY

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
D/I	AND 001	0 Z/C	← Memory Address →		
D/I	XOR 010	0 Z/C			
D/I	IOR 011	0 Z/C			
D/I	JSB 001	1 Z/C			
D/I	JMP 010	1 Z/C			
D/I	ISZ 011	1 Z/C			
D/I	AD* 100	A/B Z/C			
D/I	CP* 101	A/B Z/C			
D/I	LD* 110	A/B Z/C			
D/I	ST* 111	A/B Z/C			

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
0	SRG 000	A/B 0 D/E	*LS 000	†CLE D/E ‡SL*	*LS 000
		A/B 0 D/E	*RS 001	D/E	*RS 001
		A/B 0 D/E	R*L 010	D/E	R*L 010
		A/B 0 D/E	R*R 011	D/E	R*R 011
		A/B 0 D/E	*LR 100	D/E	*LR 100
		A/B 0 D/E	ER* 101	D/E	ER* 101
		A/B 0 D/E	EL* 110	D/E	EL* 110
		A/B 0 D/E	*LF 111	D/E	*LF 111
		NOP 000	000	000	000

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
0	ASG 000	A/B 1 CL* 01	CLE 01	SEZ SS* SL*	IN* SZ* RSS
		A/B CM* 10	CME 10		
		A/B CC* 11	CCE 11		

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
1	IOG 000	1 H/C	HLT 000	← Select Code →	
		1 0	STF 001		
		1 1	CLF 001		
		1 0	SFC 010		
		1 0	SFS 011		
		A/B 1 H/C	MI* 100		
		A/B 1 H/C	LI* 101		
		A/B 1 H/C	OT* 110		
		0 1 H/C	STC 111		
		1 1 H/C	CLC 111		
		1 0	STO 001	000	001
		1 1	CLO 001	000	001
		1 H/C	SOC 010	000	001
		1 H/C	SOS 011	000	001

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
1	EAG 000	MPY** 000	010	000	000
		DIV** 000	100	000	000
		DLD** 100	010	000	000
		DST** 100	100	000	000
		ASR 001	000	0 1	← number of bits →
		ASL 000	000	0 1	
		LSR 001	000	1 0	
		LSL 000	000	1 0	
		RRR 001	001	0 0	
		RRL 000	001	0 0	

Notes: * = A or B, according to bit 11.
D/I, A/B, Z/C, D/E, H/C coded: 0/1.
**Second word is Memory Address.

†CLE: Only this bit is required.
‡SL*: Only this bit and bit 11 (A/B as applicable) are required.

NOTES

World-Wide Hewlett-Packard Sales & Service

Call your HP Computer Specialist at any of these convenient locations:

UNITED STATES

ALABAMA
Huntsville
Tel: (205) 881-4591

ARIZONA
Phoenix
Tel: (602) 252-5061
Tucson
Tel: (602) 298-2313

CALIFORNIA
Fullerton
Tel: (714) 870-1000
North Hollywood
Tel: (213) 877-1282
Palo Alto
Tel: (415) 327-6500
Sacramento
Tel: (916) 482-1463
San Diego
Tel: (714) 279-3200

COLORADO
Englewood
Tel: (303) 771-3455

CONNECTICUT
East Hartford
Tel: (203) 289-9394
Norwalk
Tel: (203) 853-1251

FLORIDA
Ft. Lauderdale
Tel: (305) 731-2020
Orlando
Tel: (305) 841-3970

GEORGIA
Atlanta
Tel: (404) 436-6181

ILLINOIS
Skokie
Tel: (312) 677-0400

INDIANA
Indianapolis
Tel: (317) 546-4891

LOUISIANA
Kenner
Tel: (504) 721-6201

MARYLAND
Baltimore
Tel: (301) 944-5400
Rockville
Tel: (301) 948-6370

MASSACHUSETTS
Lexington
Tel: (617) 861-8960

MICHIGAN
Southfield
Tel: (313) 353-9100

MINNESOTA
St. Paul
Tel: (612) 645-9461

MISSOURI
Kansas City
Tel: (816) 763-8000
St. Louis
Tel: (314) 962-5000

NEW JERSEY
Paramus
Tel: (201) 265-5000
Cherry Hill
Tel: (609) 667-4000

NEW MEXICO
Albuquerque
Tel: (505) 265-3713
Las Cruces
Tel: (505) 526-2485

NEW YORK
Albany
Tel: (518) 869-8462
Endicott
Tel: (607) 754-0050
Poughkeepsie
Tel: (914) 454-7330
Rochester
Tel: (716) 473-9500
Roslyn, Long Island
Tel: (516) 869-8400
Syracuse
Tel: (315) 454-2486

NORTH CAROLINA
High Point
Tel: (919) 885-8101

OHIO
Cleveland
Tel: (216) 835-0300
Columbus
Tel: (614) 846-1300
Dayton
Tel: (513) 298-0351

OKLAHOMA
Oklahoma City
Tel: (405) 848-2801

OREGON
Portland
Tel: (503) 292-9171

PENNSYLVANIA
Monroeville
Tel: (412) 271-0724
King of Prussia
Tel: (215) 265-7000

RHODE ISLAND
East Providence
Tel: (401) 434-5535

TEXAS
Richardson
Tel: (214) 231-6101
Houston
Tel: (713) 781-6000
San Antonio
Tel: (512) 434-4171

UTAH
Salt Lake City
Tel: (801) 487-0715

VERMONT
South Burlington
Tel: (802) 658-4455

VIRGINIA
Richmond
Tel: (703) 285-3431

WASHINGTON
Bellevue
Tel: (206) 454-3971

CANADA

ALBERTA
Edmonton
Tel: (403) 482-5561

BRITISH COLUMBIA
North Burnaby
Tel: (604) 433-8213

MANITOBA
St. James
Tel: (204) 786-7581

NOVA SCOTIA
Halifax
Tel: (902) 455-0511

ONTARIO
Ottawa
Tel: (613) 722-4223
Rexdale
Tel: (416) 677-9611

QUEBEC
Pointe Claire
Tel: (514) 697-4232

EUROPE

AUSTRIA
Vienna

BELGIUM
Brussels

DENMARK
Birkeroed

FINLAND
Helsinki

FRANCE
Orsay
Lyon

GERMANY
Berlin W
Böblingen
Dusseldorf
Frankfurt
Hamburg
Munchen

ITALY
Milan
Rome

NETHERLANDS
Amsterdam

NORWAY
Haslum

SWEDEN
Bromma
Mölndal

SWITZERLAND
Zurich
Meyrin 2 Geneva

UNITED KINGDOM
Altrincham, Cheshire
Slough, Bucks

FOR EUROPEAN AREAS NOT LISTED, CONTACT:

Hewlett-Packard Schweiz S.A.
Rue du Bois-du-Lan 7,
1217 Meyrin 2 Geneva
Tel: (022) 41 54 00

CENTRAL AND SOUTH AMERICA

ARGENTINA
Buenos Aires

BRAZIL
Sao Paulo
Rio de Janeiro
Porto Alegre

MEXICO
Mexico City

VENEZUELA
Caracas

AFRICA, ASIA, AUSTRALIA

AUSTRALIA
Melbourne
Sydney
Adelaide
Perth, W.A.

INDIA
Bombay

ISRAEL
Tel-Aviv

JAPAN
Osaka
Nagoya
Tokyo

NEW ZEALAND
Wellington

PHILIPPINES
Manila

SINGAPORE

SOUTH AFRICA
Cape Town
Johannesburg

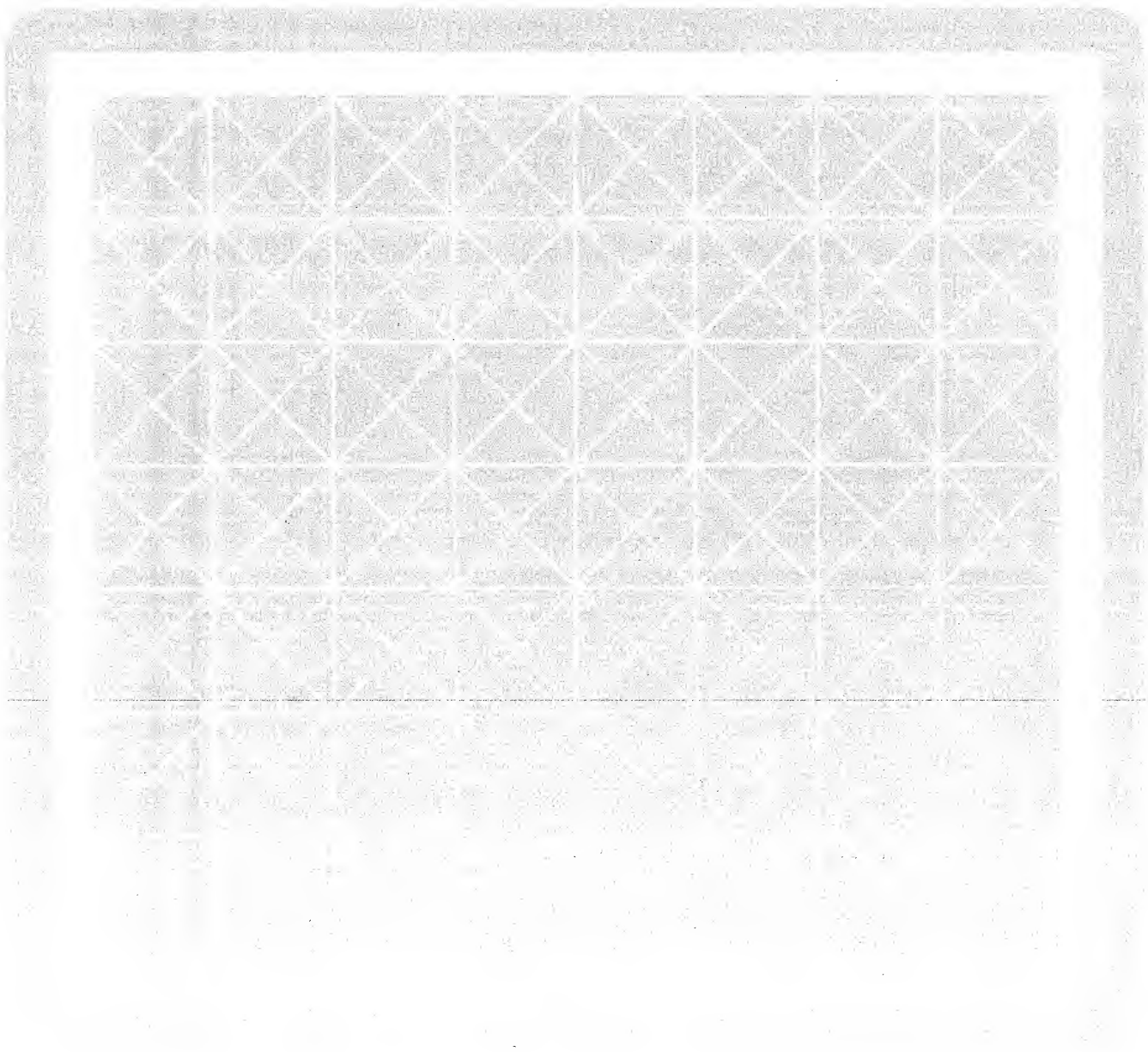
TAIWAN
Taipei

FOR OTHER AREAS NOT LISTED CONTACT:

Hewlett-Packard Intercontinental
3200 Hillview Ave.
Palo Alto, California 94304, U.S.A.
Telex: 034-8461
Cable: HEWPACK Palo Alto

Additional information on 2100A Computer software systems, peripherals, and interfaces is available from your local Hewlett-Packard Sales Office. Computer hardware manuals for the 2100A are available in three separate publications: 2100A Computer Installation and Maintenance Manual, 2100A Computer Diagrams Manual, and 2100A Computer Illustrated Parts Breakdown Manual.

HEWLETT hp PACKARD



02100-90001

Printed in U.S.A.